NANYANG TECHNOLOGICAL UNIVERSITY

SCE16-0152

WEB-BASED MATHEMATICAL DOCUMENT RETRIEVAL FOR MOBILE ANDROID APPLICATION

PREPARED BY:

DEKA AULIYA AKBAR

U1323056K

SUPERVISOR: A/P HUI SIU CHEUNG

EXAMINER: PROF. LAM KWOK YAN

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

NANYANG TECHNOLOGICAL UNIVERSITY

2017

NANYANG TECHNOLOGICAL UNIVERSITY

SCE16-0152

WEB-BASED MATHEMATICAL DOCUMENT RETRIEVAL FOR MOBILE ANDROID APPLICATION

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF BACHELOR OF COMPUTER ENGINEERING OF THE NANYANG TECHNOLOGICAL UNIVERSITY

PREPARED BY:

DEKA AULIYA AKBAR

U1323056K

SUPERVISOR: A/P HUI SIU CHEUNG

EXAMINER: PROF. LAM KWOK YAN

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

NANYANG TECHNOLOGICAL UNIVERSITY

2017

ABSTRACT

Advancement in web and mobile technology has developed innovations to human needs in various fields. One of them is to assist students learning better the subjects. Mathematics is an essential subject that can be made interesting through mobile application platform.

This project aims to enrich student's learning experience by developing a mobile Android application for a Mathematical Learning System named MathQA. The MathQA app intends to provide intuitive user interfaces for displaying educational mathematical contents and support search utilities for finding relevant mathematical documents. A mathematical document contains textual and mathematical formula information hence LaTeX typesetting is used to represent these documents.

Building a mathematical document retrieval service is non-trivial due to the highly symbolic and structured nature of the mathematical formulas. Therefore, formula-based retrieval techniques proposed in [1] and [2] were first investigated and evaluated. Based on the evaluation results, these techniques were found to have a promising performance and thus incorporated into the MathQA system.

This project can be divided into three main phases: (i) to explore mathematical document retrieval techniques, develop the database and mathematical document retrieval services in the MathQA server; (ii) to research various resources and best practices for developing web-based and android applications and incorporate them during MathQA development; (iii) to combine photographic capability in modern cameras with Optical Character Recognition (OCR) and retrieval services for finding relevant documents. Available image pre-processing and OCR tools for Android were explored to develop a reliable and robust OCR engine for recognising document images.

MathQA was developed with good design practices which follow Agile software lifecycle in order to produce a maintainable, flexible and extensible software applications. From the process and output of the project, it can be concluded that the MathQA application is useful to facilitate learning mathematics.

Keywords: Mobile Learning, Android, Django, Mathematical Document Retrieval, LaTeX, Document Image Analysis and Recognition, OCR.

ACKNOWLEDGEMENT

The author would like to express her sincere gratitude and appreciation to the following people for their constant support, encouragement, and guidance over the course of the Final Year Project (FYP):

(1) Author's supervisor, Associate Professor Hui Siu Cheung, for his invaluable advice, guidance, and criticism given throughout the course of the project. Without his support and patience, the completion of this FYP would not have been possible. The author is very grateful to be given chance to work on this fulfilling project.

(2) Mr. Le Sanh Punc, the Project Officer, for imparting his knowledge on the server-side development.

(3) Michael Jonathan Kosasih, for his assistance and cooperation during formula-based retrieval evaluation.

(4) Beloved families and friends for their continuous support and encouragement.

The author dedicates the whole FYP efforts and this report to her beloved parents, whose unconditional love, full supports and patience inspires author to work hard for the things that she aspires to achieve.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

1.1 Background

Smartphones and handheld devices have great potential in improving student's learning experience. Mobile phones have been significantly improved in these few years and provide convenience to the users with its many features and portability. The benefits of smartphones can be applied to the educational domain.

Prior to this project, a web-based learning system named MathQA was developed. MathQA system aims to provide educational mathematical content targeted for primary to junior college students. However, until now MathQA was only available as a web-based system, therefore, there is still a need to develop a mobile application into the MathQA system so that the educational contents can be accessed easily by students from anywhere at any time.

Developing a mobile application for mathematical documents is not trivial. First, the mathematical domain of the MathQA system implies that the documents in the system involve textual and mathematical formula contents. Mathematical contents are different from the textual contents because they may contain complicated symbols and embedded with semantic and structural meanings.

Another problem is that there were no mathematical document retrieval services available in the current MathQA system, which makes the system extremely impractical. Therefore, robust mathematical document retrieval techniques must be investigated and built into the MathQA system to allow students to find mathematical problems easily and immediately. However, the complex nature of mathematical documents which mix textual and formula content may lower the retrieval accuracy. Therefore, in this project, the mathematical document retrieval techniques are separated into two parts: text-based and formula-based retrieval.

Last, author identified that typing queries for searching a mathematical content is tedious. However, now that most smartphones are equipped with high photographic capability, it opens a new opportunity for Optical Character Recognition (OCR) features to be complemented into the mobile application. The OCR can first recognise

textual content from a document image, and users can then use the OCR result to perform mathematical document retrieval with the MathQA system.

1.2 Project Objectives

This project aims to develop a mobile application for learning mathematics. In doing so, the Android app must able to display mathematical contents from the server and perform a mathematical document retrieval to the server. Mathematical document in this project refers to mathematical questions available in the MathQA database.

Based on these objectives, the project development stage is divided into two main parts:

(1) Web server development which provides the educational mathematics contents and search services for retrieving mathematical documents.

(2) Android application development which provides user interface (UI) for accessing and searching mathematics contents. To provide convenience to the user, OCR service is incorporated into the app to allow users performing a search through document image recognition.

1.3 Project Overview

Figure 1 depicts the overall architecture of the MathQA system for this project. The system is composed of two main components: the web server and android app.

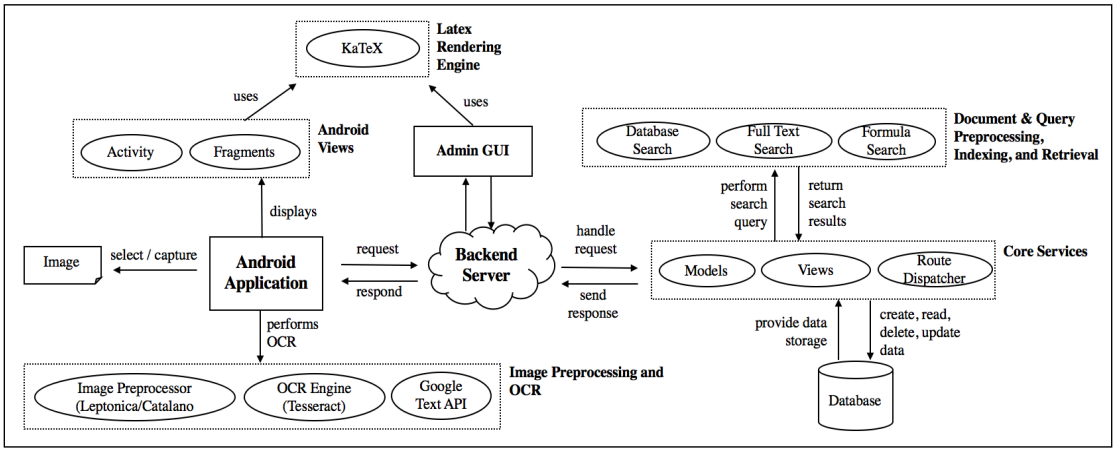


Figure 1. Overall Architecture

The web server component provides services for accessing mathematical content from the database and search services for retrieving mathematical documents. The data used to build these services was collected from previous students' work, which involves mathematical question data from O-level, A-level, and PSLE papers over the

period of 1995-2010. Three types of mathematical document retrievals were implemented in the system, which includes database, text-based and formula retrievals. Along with the server development, an admin GUI was also developed to help analyse the LaTeX contents.

For Android, an intuitive user interface (UI) for displaying mathematical documents was designed, prototyped, and developed into a working app. The mobile app allows users to access and search for mathematics content from MathQA through MathQA server's REST API. Author then researched about OCR implementation for mobile app, and incorporated this technology onto the app. Users can benefit from this OCR feature because it allows them to search for a mathematical problem without having to type in their queries.

1.4 Project Scope

The scope of the project encompasses the following work:

(1) Design and develop a database model for providing educational mathematical contents.

(2) Clean the mathematical database contents from previous work to fix the syntax errors and standardisation for LaTeX. Mathematical contents, in this case include textual and LaTeX contents. For this objective, admin GUI that can help analyse mathematical contents in the database is needed.

(3) Investigate approaches to perform mathematical document retrieval and implement mathematical document retrieval techniques to find the most relevant document for a given user's query.

(4) Provide REST APIs to allow the mobile app to connect with the MathQA services.

(5) Design and develop an android mobile application that allows LaTeX rendering and therefore able to display the mathematical contents from MathQA database.

(6) Investigate current approaches and open-source technologies for Optical Character Recognition in a mobile application platform.

(7) Design and develop an Android mobile app that can recognise text from a captured image by utilising open-source OCR technologies that involve pre-processing, processing and post-processing of the document image.

1.5 Project Schedule

Table 1 depicts the timeline activities that the author had done during FYP. In the first semester, the author focused more on gathering information and knowledge related to the FYP problem. This includes gathering use case requirements, project planning, and explorations on the related Android and web-based technologies. During the first half of the project, the author identified the main functionalities of the app, decomposed these into smaller functional unit and performs exploratory work on one functional unit at a time.

Table 1. Final Year Project Schedule

Activity	Semester 1 – 16/17												Semester 2 - 16/17					
	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC	JAN	FEB	MAR	APR	MAY					
Project Selection and Introduction																		
Familiarise with Related Projects and Web-Mobile App Development																		
Brainstorming for System Requirements																		
Project Planning and Proposal																		
Requirement Elicitations and Analysis																		
Initial System Design and Architecture																		
Design Lo-fi Prototype for Android																		
Develop Hi-fi Android prototype and exploring related open source tools and libraries																		
Familiarise with Mathematical Document Retrieval																		
Familiarise with the new MathQA Server																		
Full Server Development																		
Full Android Development																		
System Integration																		
Testing and Evaluation																		
Final Report Writing																		
Report Amandement																		
FYP Oral Presentation Preparation																		

At the end of the first semester, the author met with the Project Officer and was introduced to the new MathQA system. The semester break was then spent to familiarise with the new system environment and exploring mathematical document retrieval based on Ma Kai's [1] and Sidath's works [2]. The proposed mathematical document retrieval techniques proposed are incorporated into the MathQA system during the server development. The main full server and Android development were started from the semester break onwards, by integrating all the related knowledge, appropriate tools and libraries that the author had acquired from the exploratory work done in the first semester.

1.6 Report Organisation

Table 2 describes the organisation of the report.

Table 2. Report Organisation

Chapter	Description
1. Introduction	Discusses the background and motivation of the project, the project objectives and the project scope.
2. Related Work	Highlights the background information related to the project. It introduces mathematical documents, web-based information retrieval, related work on the mathematical document retrieval, Android device, document image analysis and recognition, and software engineering concepts.
3. Project Overview	Describes the software engineering approaches used during project development, including the project requirements, system design and architecture and database design.
4. Web Server Development	Discusses about the architecture, related tools, libraries, and implementation details for web server development.
5. Mathematical Document Retrieval	Describes approaches taken to implement Mathematical Document Retrieval for the back-end service. Relevant experiments conducted to evaluate the retrieval service were discussed in this chapter.
6. Android Development	Discusses about the architecture, related tools, libraries, and implementation details for Android app development.
7. Document Image Analysis and Recognition	Describes the approaches taken to implement OCR on Android. Relevant experiments conducted to evaluate the OCR performance were discussed in this chapter.
8. Conclusion and Future Works	Concludes the project and suggests recommendations for future works.

2 RELATED WORK

This section highlights relevant information and related works that were used during the project development.

2.1 Mathematical Document

A mathematical document is a document that has both textual and mathematical contents. To represent mathematical content in its highly symbolic and structured form, a mathematical representation is required. Two mathematical representations - LaTeX and MathML- are utilised throughout the project for rendering mathematical content and mathematical document retrieval purposes.

2.1.1 LaTeX

LaTeX is a document preparation system for high-quality typesetting and a document mark-up language [3]. LaTeX supports typesetting of complex mathematical formula which is widely used to represent mathematical formulas in the web. Because of this reason, the web server development and Android development will use LaTeX as the proper mark-up language for representing mathematical formulas.

LaTeX declared special environments for displaying mathematical elements. This is required because mathematical notations are displayed differently from normal text. These environments include:

a. *Inline*: formulas are displayed within the text body. To declare inline formulas, the formula needs to be surrounded by a single dollar sign or backslash followed by bracket:

```
$latex_formula$ or \(latex_formula\)
```

Figure 2. LaTeX Syntax in Inline-mode

b. *Displayed*: formulas are displayed separately from the text body. To declare displayed formulas, surround the latex formula with double dollar signs or backslash followed by a square bracket.

```
$$latex_formula$$ or \[latex_formula\]
```

Figure 3. LaTeX Syntax in Display-mode

2.1.1.1 LaTeX Rendering

LaTeX requires TeX rendering engine to display mathematical contents. As the mobile and web-based applications that run on HTML, JavaScript, and Java; TeX rendering engines based on these languages were explored. After some explorations, KaTeX, a math TeX rendering library was found to be the best LaTeX rendering engine. The reason for this is because it is very fast and light weighted compared to other LaTeX engine libraries. Therefore, KaTeX is used in both web-based and mobile application development to render LaTeX string.

2.1.2 MathML

Mathematical Mark-up Language (MathML) is a mathematical mark-up language, an application of XML for describing mathematical notations that captures both its structure and content [4]. Although LaTeX is another mark-up language, it only provides a typesetting tool, hence it does not embody the relationship between operators, constants, and variables from the formula. For this reason, MathML is used to capture the semantic and structural meaning of the formulas.

Table 3. Comparison between LaTeX, rendered LaTeX and MathML representation of the same Sample Equation

LaTeX Representation	Rendered Formula (LaTeX)	MathML Representation

`$$\int 5e^{2x-1} {d}x$$`	$\int 5e^{2x-1}dx$	`<math>`
		`<mrow>`
		`<mo>∫</mo>`
		`<mn>5</mn>`
		`<msup>`
		`<mi>e</mi>`
		`<mrow>`
		`<mn>2</mn>`
		`<mi>x</mi>`
		`<mo>−</mo>`
		`<mn>1</mn>`
		`</mrow>`
		`</msup>`
		`<mrow>`
		`<mi>d</mi>`
		`</mrow>`
		`<mi>x</mi>`
		`</mrow>`
		`</math>`

2.2 Web-Based Information Retrieval

2.2.1 Information Retrieval

Information retrieval (IR) is finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections [5]. In this project, the vector space model is adopted to represent

documents and queries as set of vectors of index terms. This vector can be implemented using an inverted index data structure. An inverted index is an index table that maps back from the terms to the parts where these terms occur in the document.

A scoring scheme, such as Term Frequency-Inverse document frequency (TF-IDF), is usually used to compute the similarity between the query and the documents for ranking the document results. TF-IDF weight evaluates how important a term is in a document or corpus.

$$\text{idf}_t = \log \frac{N}{\text{df}_t}$$

$$\text{tf-idf}_{t,d} = \text{tf}_{t,d} \times \text{idf}_t$$

N: total number of documents in the collection

DF_t: Number of documents that contain the term

Figure 4. TF-IDF Formula

2.2.2 Web

The web is a jargon that can refer to one of the following things: 1) Web Page: A document which can be displayed in a web browser. A web page is usually written in HTML, styled with CSS and controlled with JavaScript. 2) Website: A collection of web pages which are grouped together and connected in various ways, 3) Web server: A computer that hosts a website on the Internet, and 4) Search engine: A website that helps find other web pages [5]. In the scope of this project, MathQA system is a web server which hosts web pages that provide assisted learning tools to help students understanding mathematical problems. A search engine for searching mathematical formulas and questions will be built into the system.

The following points are some technical terms related to the MathQA server development.

2.2.2.1 Application Programming Interface (API)

API is a set of functions, routines, protocols and tools for building software applications which access the features of the data of an operating system, application, or other services.

2.2.2.2 Representational State Transfer (REST) API

REST is a web architecture style that regulates the behaviour between client and server. REST APIs define a set of functions from which developers can request and receive responses via HTTP protocol such as GET and POST.

Table 4. Example of a REST API

HTTP Request	Action
`GET` http://localhost:8000/api/formulas/	Retrieves existing formulas from MathQA database

2.2.2.3 Django

Django is a free and open-source high-level web framework for Python. Django follows a model-view-pattern architectural pattern and provides set of ready-made components or modules that simplify the web development [6]. Django offers powerful object-relational mapping (ORM) which handles various database operations, such as Create, Delete, Update, and Delete (CRUD) as well as advanced querying. This is made possible through Django's queryset APIs, which abstracts SQL database queries and allows database operations to be written in plain Python language.

2.2.2.4 AngularJS Material

AngularJS is an open-source front-end JavaScript framework developed by Google. This framework offers prominent features such as two-way data bindings, controllers and directives which aim to simplify and accelerate the web app development process. Angular Material is a UI component framework that implements Google's Material Design Specification for AngularJS. This library provides set of reusables, well-tested, and accessible UI components based on Material Design [7].

2.2.3 Web Search

A web search is a system designed to look for information on the internet. A search engine consists of three components: *Spider*, which builds the corpus; the *Indexer*, which creates inverted indexes by first normalising the corpus; and *Query* processor which process the query and serve the query results [8]. The following steps were taken to develop the mathematical document retrieval system in this project:

(1) Build a corpus by performing term extractions on both text and formula content of the mathematical questions.

(2) Create an inverted index table for both formula and textual contents of a question.

(3) Pre-process user's query before finding the relevant mathematical questions in the server.

(4) Compute similarity scores between query and documents.

(5) Return the search results sorted in decreasing order of similarity score.

2.3 Mathematical Document Retrieval

In the MathQA system, a document is defined to be a mathematical question. A mathematical question can contain textual and formula contents. In this project, mathematical document retrieval is separated into textual and formula search. The reason of this separation is because textual and formula information have different embedded meanings. The core meaning of text documents are composed of word units, while formulas are composed of symbols, numbers, variables and structural relationships between its constituents. Therefore, a typical text-based retrieval approach cannot be applied to mathematical formulas. For this reason, a different approach to perform formula-based retrieval was investigated.

2.3.1 Text-based Retrieval

Text-based mathematical document retrieval implementation in this project uses the inverted index technique. The word terms in the question content are pre-processed and stored as the key in the inverted index table that maps back to the list of questions that contain that pre-processed terms. When the server receives a query, the query is first pre-processed, then the terms are compared with the relevant documents in the inverted-index table, the tf-idf score between query and math questions are computed and finally, the list of relevant math questions will be returned to the users in descending order of tf-idf scores.

2.3.2 Formula-based Retrieval

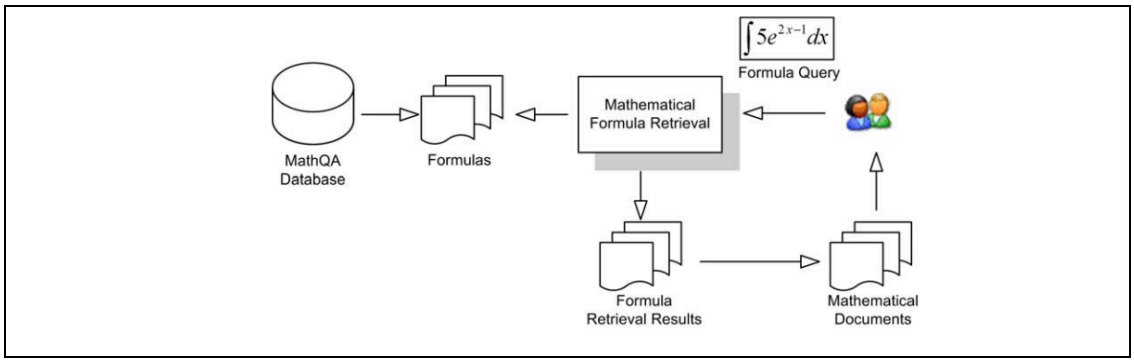


Figure 5. Mathematical Formula Retrieval (Ma Kai)

Formula retrieval technique presented in the paper "*Data Mining for Mathematical Question Answering Community*" by Ma Kai (2011) is adopted as a reference for building and evaluating a formula search system. In the paper, in order to perform a formula search, a formula processing layer needs to be implemented on top of a typical search engine. The role of this formula processing is to extract formula terms from a LaTeX string. These formula terms are equivalent to the word terms in the textual document, and therefore can be indexed into an inverted index table [1].

2.3.2.1 Formula Features Extraction

From the paper, every formula has five terms, they are:

(1) Semantic Features, which capture semantic information in the formula. This feature includes mathematical functions and operators. The semantic feature of a formula is divided into two types, *in-order* and *sorted* terms. Semantic in-order terms are the 2, 3, 4-grams of the original ordered sequence of semantic features while the sorted terms are the lexicographically sorted semantic feature sequence.

(2) Structural Features, which capture the structural relationship between elements in the formula.

(3) Constant Features, which capture the number or constant information in the formula.

(4) Variable Features, which capture the variable information in the formula.

During the formula retrieval process, in-order terms, sorted terms, structural features, constant and variable features are first extracted from the query and then compared with the existing formula terms of the formulas available in the database.

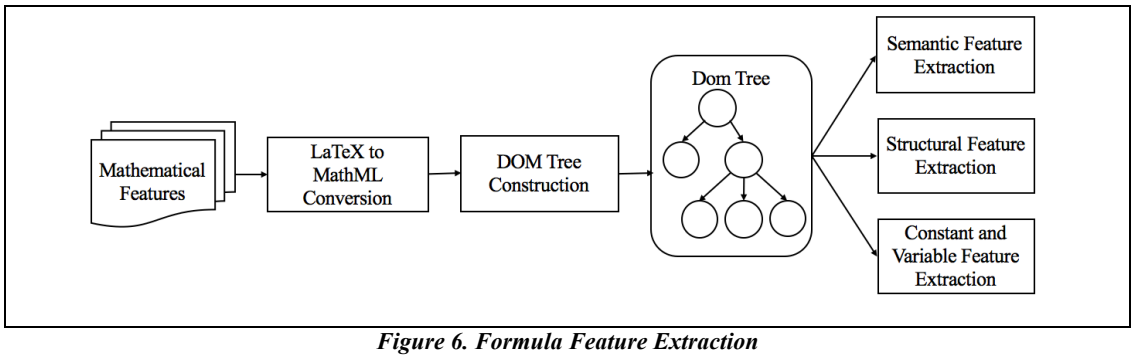


Figure 6. Formula Feature Extraction

2.3.2.2 Formula Indexing

Prior to formula retrieval, a formula index table must exist. This table is created by storing the formula terms of all available formulas as inverted indexes. The formula terms available in step 2.3.2.1 are first generated. Each formula term then becomes

the index of an entry in the FormulaIndex table where the content of each entry list of formulas that contain this formula index term. The document frequency of a formula term is also sorted in the table to allow performing a matching scoring computation.

2.3.2.3 Formula Retrieval

This step aims to find the most similar formulas to the formula query using the inverted index technique. This step is dependent on step 2.3.2.2. Therefore, prior to the formula retrieval, a formula index table must first exist. The formula retrieval involves three main processes: query processing, related formula retrieval and ranking. Query processing is performed on the query to generate formula query terms using the same step as section 2.3.2.1. After the query terms are generated, the formulas related to the query are retrieved. Related formulas are defined to be those formulas that have at least one matching formula term with the query. However, this operation may lower the retrieval efficiency especially when the number of matched formulas is very high. Therefore, the top-K retrieval technique is adopted to solve this problem. In this technique, a candidate limit K is set, so as soon as the number of candidates reaches K, the search results can be returned.

2.3.2.4 Formula Ranking

Formula ranking is required to present the search results in the order that is relevant to the user's query. Similar terms are first computed by applying set intersection and set difference operations between the formula and the query. Formula ranking is then calculated by computing the similarity score between related formulas and the formula query using IDF weighting scheme.

$\mathrm{IDF}(t) = \log \frac{N}{\mathrm{DF}(t)}, \quad \mathrm{DF} \neq 0$	N: #formulas in the collection DF(t): #formulas in the collection that contains formula term t

Figure 7. IDF Weighting Computation for Formula Terms

The weighting scheme used during the overall score computation assigns more importance to the semantic and structural features than constant and variable features. Functional operators (e.g.: sin, cos, log) are also given more weight than the symbolical operators (e.g.: +, x, /). After the ranking, the related formulas will be sorted in decreasing order of the similarity score. Therefore, the position of the formula in the results indicates the formula relevance with the formula query.

2.3.2.5 Mathematical Document Retrieval

From the ranked formula results, the questions that contain these ranked relevant formulas were returned to the users. In the database design, a Many-to-Many relationship is established between formula and question. Therefore, retrieving questions from the related formulas is straightforward.

2.4 Mobile Device

Nowadays, handheld devices, such as smartphones are very powerful in terms of computing power, high-quality camera and communication services that allow data to be transmitted over mobile networks. All these features have enabled developers to develop a mobile application which makes use of the highly available resources in the mobile device to provide an educational platform that is engaging and enriches the learning experience for students.

2.4.1 Android Background

Android is an open-source mobile operating system developed by Google. It is a full software stack for mobile devices based on Linux kernel which includes an operating system, middleware and key applications. Android applications are written in Java and run on Dalvik, a virtual machine designed for embedded use that runs on top of a Linux kernel [9]. Below are some technical terms associated with Android development.

2.4.1.1 Activity

An activity is a single, focused thing that the user can interact with. Almost all activities interact with the user. Therefore, the Activity class takes care of creating a window for developers to place the mobile application user interface (UI) [10].

2.4.1.2 Fragment

A fragment represents a portion of UI in an Activity. The fragment is a modular section of an activity, which has its own lifecycle, receives its own input events, and can be added or removed while the activity is running [10].

2.4.1.3 View

This class provides the basic building block for UI components. A View occupies an area on the screen and responsible for drawing and event handling. The view is the base class for Android widgets, which are used to create interactive UI components [10].

2.4.1.3.1 RecyclerView

RecyclerView is a flexible view commonly used as a container optimised for rendering large data set. It is a flexible and efficient version of ListView (a view that displays a list of items). In this project, RecyclerView is used to display a list of data contents received from MathQA server.

2.4.1.3.2 ViewPager

ViewPager is a layout that allows users to swipe left and right through "pages" of content which are usually different fragments in an Activity.

2.4.1.4 UI and Background Thread

By default, an Android app runs on a single thread (the “main” or “UI” thread). The role of the main thread is to handle events and user interactions with the views of the UI. Most of the additional app components and tasks will run in the main thread by default. However, when a time-consuming task exists, using the main thread creates an adverse effect where it may cause the entire application to be unresponsive until the task is completed [11]. To solve this problem, heavy tasks need to run on a separate thread so that the main thread can continue to run unobstructed by other tasks. In this project, heavy tasks such as retrieving data from the server, storage access, image pre-processing and image recognition were run on separate threads. A progress indicator was also developed to provide feedback regarding the current status of the app.

2.4.1.5 Material Design

Material Design is a visual design language specification developed by Google. It is a comprehensive guide for visual, motion, and interaction design across platforms and devices. Material Design specification is followed to develop a good Android application UI.

2.5 Document Image Analysis and Recognition

Document image analysis analyses the document images to extract the text and graphics information from an image. Printed character recognition is important in the context of document image analysis [12].

2.5.1 Image Pre-processing

Image pre-processing is the process of adjusting digital images so that the results are more suitable for display or further image analysis.

2.5.1.1 Image Enhancement

During this pre-processing, techniques such as brightness and contrast normalisation were applied.

2.5.1.2 Grayscaling

Converts a true color image from a RGB (red, green, blue) scale into grayscale by eliminating the hue and saturation information while retaining the luminance. The grayscale intensity is stored as an 8-bit integer giving 256 possible different shades of gray from black to white [13].

2.5.1.3 Image Binarisation

Image binarisation is a type of image segmentation method. In this step, a grayscale image is reduced into a binary image of text (black) and background (white). Binarisation is required to isolate candidate text regions and provide a sharp contrast with the background. Binarisation is commonly applied using thresholding, a technique to partition the colors in the image into two sets based on a histogram or other statistical measure [14]. In this project, two binarisation techniques were experimented using Otsu and Sauvola. From the observed results, it is found that binarising document image using Sauvola's method preserved more legibility.

2.5.1.4 Image De-skewing

Image de-skewing attempts to correct the document image alignment. A correctly aligned document simplifies the character and text line detection algorithms [15]. There are two steps in image de-skewing process. First, skew detection is performed to detect the skew angle of a document image. Both pre-processors used in this project used Hough line transform [16] technique to detect the skew angle of a binarised image. Second, the computed skew angle is then used to perform an image rotation in the opposite direction, therefore the skew of an image can be corrected.

2.5.2 OCR

Optical character recognition (OCR) systems enable an automatic pattern recognition of alphanumeric and handwritten characters in scanned documents or images. The followings are the available technologies related to OCR that are found to be compatible for Android environment.

2.5.2.1 Tesseract

Tesseract OCR engine is an open-source OCR engine developed in HP between 1984 to 1994 which put emphasis on line finding, features/classification methods and adaptive classifier [17].

2.5.2.2 Leptonica and Catalano

Leptonica [18] and Catalano [19] are examples of open-source libraries for image processing and image analysis applications. These two libraries were used to perform image pre-processing prior to text recognition using an OCR engine. Leptonica, in particular, is recommended by Tesseract developers to perform image processing.

2.5.2.3 Google Text API

Google Text Recognition API is a part of Google Vision API open-source technology developed by Google. This framework allows users to recognise text from a document image without having to pre-process the image [20].

2.6 Software Engineering Concepts

2.6.1 Agile Software Development Lifecycle

Software development lifecycle (SDLC) defines approaches and processes to ensure the success of producing high-quality software that meets user requirements. An SDLC describes a number of work phases in software lifecycle and the order of how these phases are executed. For this project, the Agile SDLC model is adopted. Agile SDLC is a combination of iterative and incremental process models that focus on providing flexibility and customer satisfaction through iterations of product deliveries.

2.6.2 SOLID

S.O.L.I.D. is an acronym for the first five Object Oriented Design principles proposed by Robert C. Martin [21]. The principles allow developers to create software applications that are maintainable and extensible, avoid code smells, code that is easily refactored, and participate in the agile or adaptive software development. The principles of object-oriented class design are as follows.

(1) Single responsibility principle: there should never be more than one reason for a class to change. This principle encourages the separation of responsibilities rather than having a master class that can do everything.

(2) Open/closed principle: software entities (class, modules, functions) should be open for extension, but closed for modification. It encourages the use of abstraction and polymorphism. A module can be closed for modification if it depends on an abstraction. The behaviour of the module can be extended by creating new derivatives of the abstraction.
(3) Liskov substitution principle: subclasses should be substitutable by their base classes. It introduces the concept of design by contract, a derived class should have the same contract as the abstract base class.
(4) Interface segregation principle: many client specific interfaces is better than having one general purpose interface.
(5) Dependency injection principle: depend upon interfaces or abstract classes, rather than concrete functions or classes.

2.6.3 Don't Repeat Yourself (DRY)

This principle states that every piece of knowledge must have a single, unambiguous, authoritative representation within a system [22]. It aims to reduce repetition of all information of all kinds.

3 PROJECT OVERVIEW

During the project development, the functionalities available in both server and Android are divided into small functional units. Then, for each functional unit, an iterative approach will be taken to build the unit into a working software following the Agile Software Development Lifecycle. All these units are then integrated together into a bigger unit incrementally, and at the final stage of the integration, a whole MathQA system composed of a web server and Android app is developed.

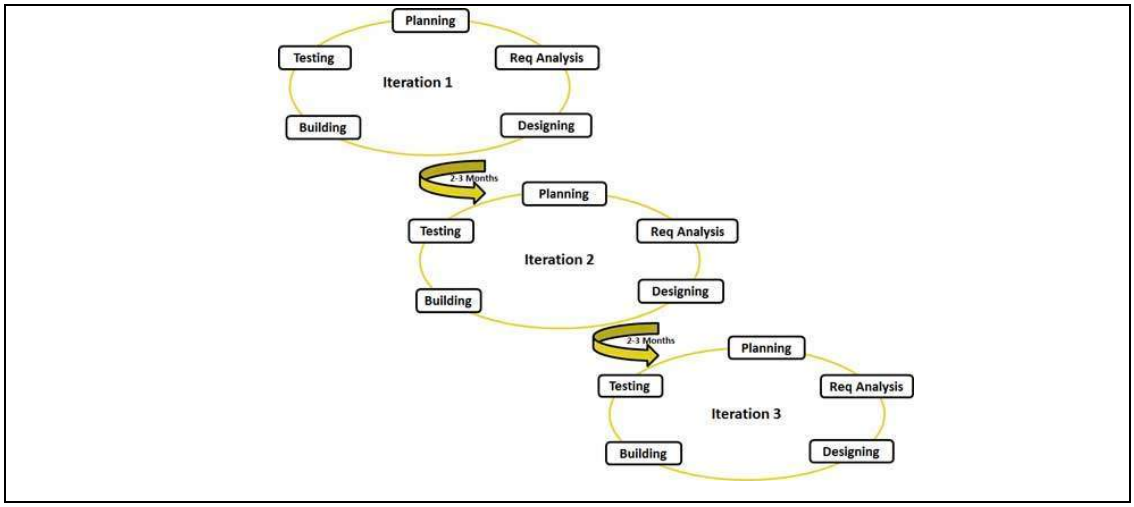


Figure 8. Agile Software Development Life-cycle

3.1 Requirements Elicitations and Analysis

Requirement analysis is the first stage of a software lifecycle. At this stage, several discussions and problem explorations based on project scope and objectives were conducted to decide the MathQA main functionalities.

3.1.1 User Classes

Two types of users are identified for the MathQA system: public user and admin. Public users only have read-only access, therefore they can only view mathematical contents from MathQA server. An authenticated admin on the other hand, has all-access to database manipulation with the MathQA system. The actions allowed for both public user and admin are available in the form of use case diagrams in Figure 9 and Figure 10 respectively.

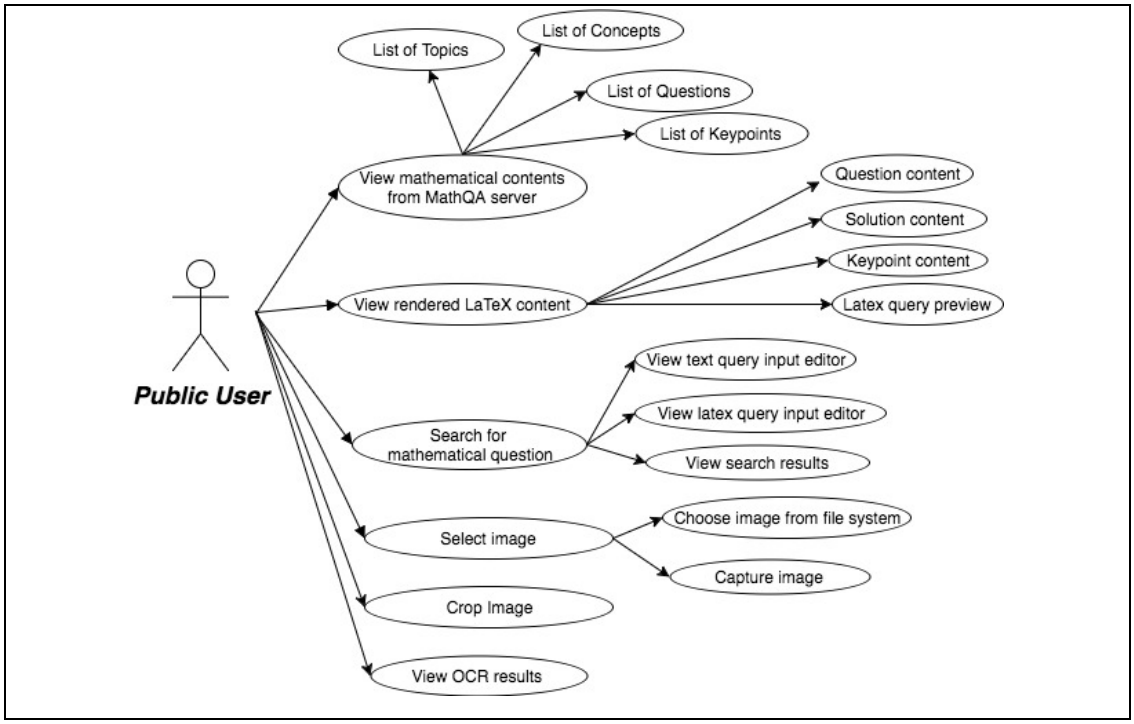


Figure 9. Use Case Diagram for Public User

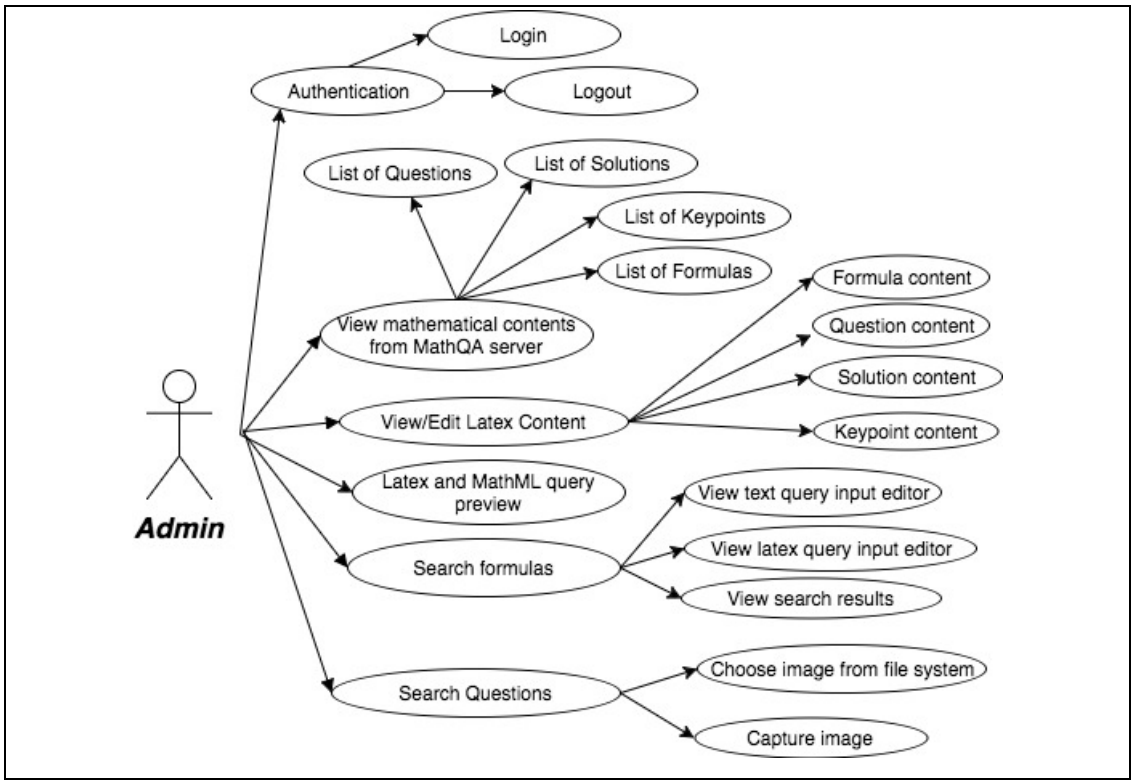


Figure 10. Use Case Diagram for Admin

3.2 Functional Requirements

From the project scope, two main components are identified in the MathQA system: the server and the Android app. Based on this system component

identification and the initial use case analysis, the functional requirements were then elicited for both server and Android components.

3.2.1 Web Server Requirements

Table 5. Web Server Requirements

Backend	Front-end
(1) Receive request from client and send back JSON responses. (2) Provide an interface for CRUD operations with the database. (3) Provide a common interface for retrieving objects with addition of queryset filters. (4) Provide logic behind mathematical document retrieval mechanism including text-based retrieval using database and full-text search, and formula retrieval.	(1) Display rendered LaTeX contents. (2) View available LaTeX contents from database. Object models that contains LaTeX contents in the Database are Questions, Formulas, Keypoints, and Solutions. (3) View and edit LaTeX content of an object instance. (4) Create new LaTeX formulas into the database and generates a MathML representation of the LaTeX syntax for debugging purpose. (5) View list of available formulas in the database as a table. (6) Perform formula search and formula filters by formula categories. (7) Reindex formulas. (8) Login authentication.

3.2.2 Android Requirements (Public user)

Table 6. Android Requirements (Public User)

Background Service	User Interface
(1) Camera access (2) Read/write memory (3) Access network (4) Receive data (5) Post data (6) Pre-process an Image	(1) View list of topics (2) View a concept (3) View list of questions (under a topic, under a concept, under a subject) (4) Enter a text query

(7) Recognise characters from an Image (8) Render LaTeX (9) Access MathQA server through REST APIs.	(5) Enter formula (LaTeX query) (6) Capture an image (7) Crop an image (8) Display data from the server (9) Render LaTeX in mathematical document from the server (10) Submit a search query by manual keyboard entry, editable OCR text result, and interactive LaTeX editor. (11) Display LaTeX (12) Display mathematical documents in logical groupings and clear manner (13) Captures image and recognise its textual content (14) Performs question search to the MathQA server

3.3 System Design and Architecture

3.3.1 Overall Architecture

Figure 1 in chapter 1 depicts the overall architecture for developing a web-based and Android MathQA applications. This architecture is referred throughout the course of the project development in order to build web services and Android app for MathQA.

3.4 Database Design

Figure 11 shows the ER diagram for the database design used in the MathQA server.

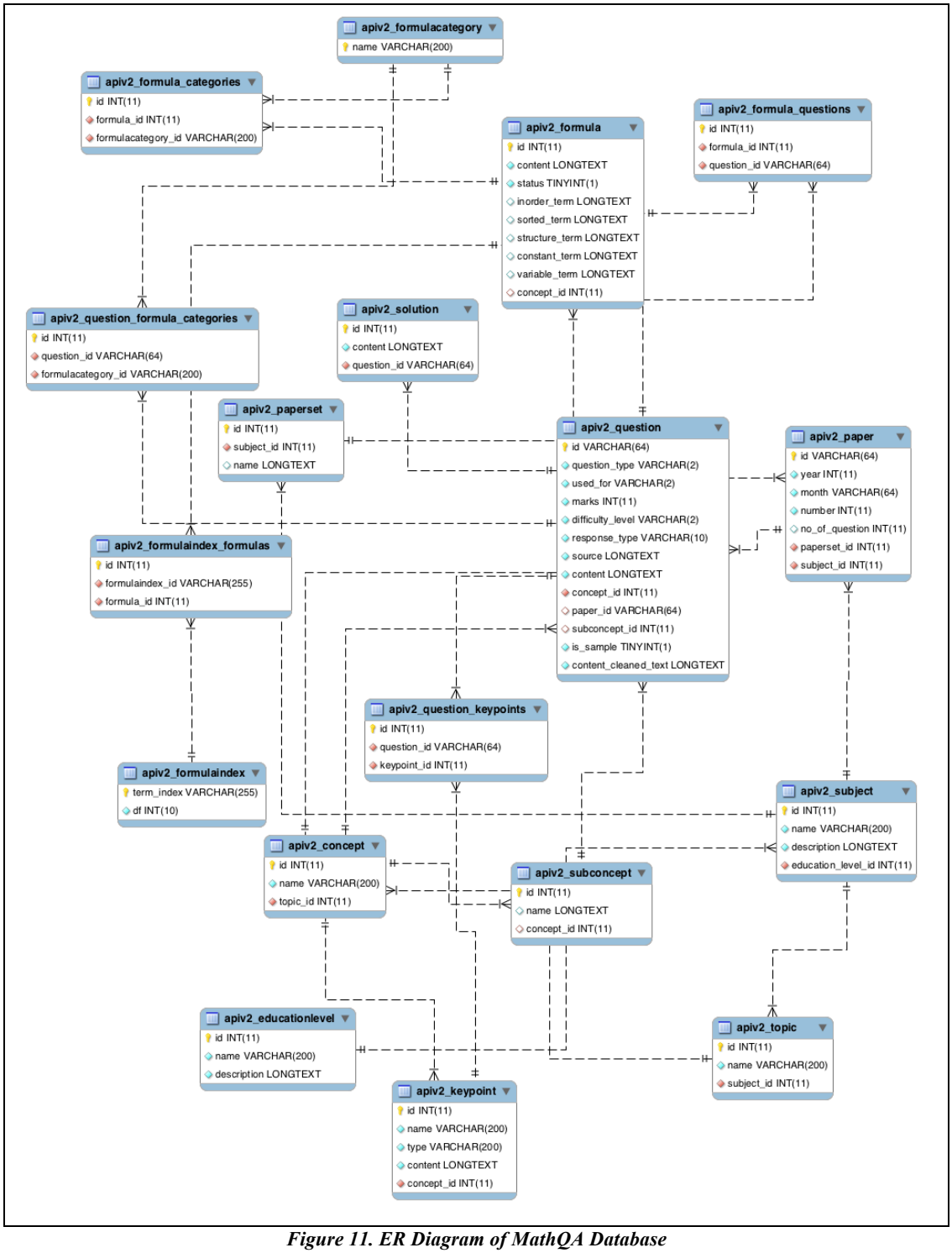


Figure 11. ER Diagram of MathQA Database

3.4.1 Database Models

The data source for developing MathQA system was gathered from previous student's work with additional modifications and adjustments to cater with the new MathQA system database models. These models are explained as follows:

3.4.1.1 Subject

The mathematical contents in the database are grouped into 4 subjects: Additional Mathematics, Elementary Mathematics, H2 Mathematics and PSLE Mathematics.

3.4.1.2 Topic, Concept and Subconcept

Each subject contains several mathematical topics. A topic can contain several concepts, and each concept is composed of several sub concepts. There are 36 topics, 81 concepts, and 264 subconcepts available in the database

3.4.1.3 Keypoint

Each concept in the database may have several key points. Key points are mathematical content that explains a mathematical theory or formulas related to a concept. A keypoint contains textual and LaTeX content. There are 458 keypoints available in the database.

3.4.1.4 Question

A question has textual and formula information. Question data are mathematical questions obtained from the set of examination papers between 1997-2010. In the database, two types of question's content are stored: full content and text only content. The full content field stores both question's textual and LaTeX contents, while for the text only content, only pre-processed text-only information is stored. This cleaned content is created to improve text-based retrieval accuracy. The question also has other properties such as the list of formulas available in the question, the maths sub concept and concept of the question, the paper where the question was obtained, and its difficulty level. There are 2163 mathematical questions available in the database.

3.4.1.5 Formula Category

Formula category groups formulas into several categories. There are 13 categories available as shown in Figure 12. These categories are formulated based on the common latex terms that usually appear in the formula. Therefore, regular expressions are applied to match formula patterns and assign formula category to the question and formula data.

```
FORMULA_PATTERN = {
    ABSOLUTE: [r'\|'],
    DIFFERENTIATION: [r'\\mathrm', r'd{.+}'],
    EXPONENTIAL: ['e\^\{[a-zA-Z]+\}', 'e\^[a-zA-Z]+', r'\d+\^\{[a-zA-Z]+\}',
                  r'\d+^[a-zA-Z]+'],
    FRACTION: [r'\\frac', 'frac'],
    INEQUALITY: ['<', '>', r'\\leq', r'\\le', r'\\geq', r'\\ge'],
    INTEGRAL: [r'\\int'],
    LINE_EQUATIONS: [r'y\s*=\s*'],
    LOGARITHMS: [r'\\log', r'\\ln', r'\\lg', 'log', 'ln', 'lg'],
    POLYNOMIAL: [r'[a-zA-Z]\^'],
    SERIES: [r'\\sum'],
    LIMIT: [r'\\lim'],
    TRIGONOMETRY: [r'\\sin', r'\\cos', r'\\tan', 'sin', 'cos', 'tan',
                   r'\\sec', r'\\cot', r'\\csc', 'sec', 'cot', 'csc'],
    SURDS: [r'\\sqrt', 'sqrt'],
}
```

Figure 12. Formula Categories

3.4.1.6 Formula

A formula is a mathematical content written in LaTeX. In order to build a formula-based retrieval, formula terms must exist first in the database. Therefore, a formula will have a LaTeX content, and the five formula terms generated in section 2.3.2.1: *in-order semantic* terms, *sorted semantic* terms, *structural* terms, *variable* terms, and *constant* terms. Formula categories are also assigned to each formula to provide a logical formula grouping.

Each formula can be linked to one or more questions; therefore, it forms a many-to-many relationship. This question-formula relationship is important during formula-based retrieval because at the post-query processing stage, relevant formulas will be returned together with the questions that contain these formulas.

Formula data is inserted manually by the author. This stems from the fact that the original data contains a lot of LaTeX syntax errors, non-standardised formulas and non-meaningful formulas such as currency (e.g.: $, £) or degree (°) symbols. The formula that is available in the database must be meaningful formulas that have been cleaned from syntax errors and standardised. Therefore, formula cleaning and standardisation step are very crucial during formula data creation.

3.4.1.7 Formula Index

FormulaIndex table plays an important role in formula retrieval. Each key in this table is a formula term which maps to a set of formula objects which contain this key term. The number of formula objects of a formula term is also stored as the document frequency field (df). This df field will be used for tf-idf similarity score computation.

3.4.1.8 Solution

Solution is the solution to a mathematical question. It contains both textual and LaTeX information.

3.4.1.9 Paper and Paperset

These two objects refer to the exam paper and the paperset where the mathematical questions were obtained.

3.5 User Interface State Machine Diagram

From the use case and functional requirements, then these requirements are modelled into a state machine diagram. These diagrams are useful for building user interface (UI) components. These diagrams are first translated into Lo-Fi UI prototypes that will eventually be turned into real web front-end and Android UI.

3.5.1 Server Front-end

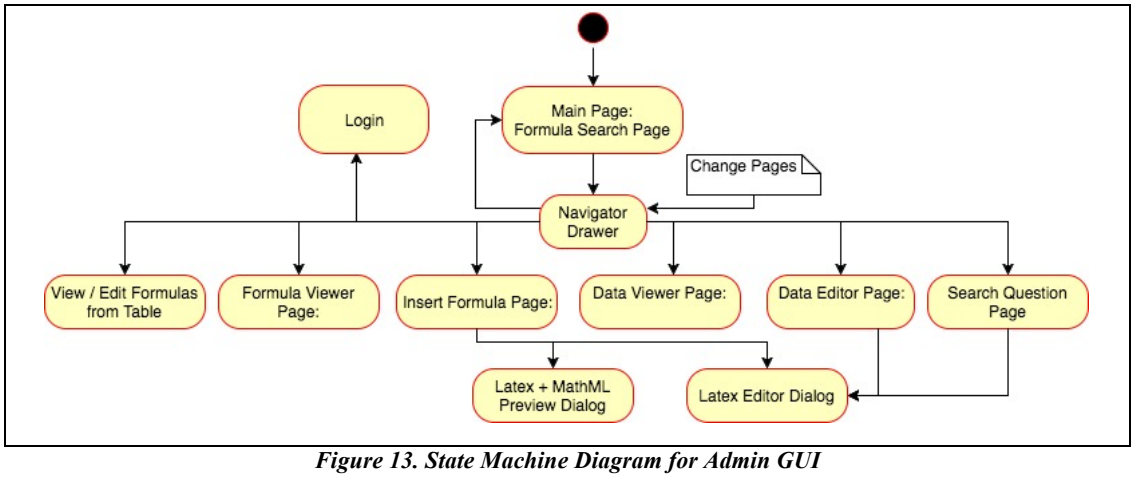


Figure 13. State Machine Diagram for Admin GUI

3.5.2 Android Front-end

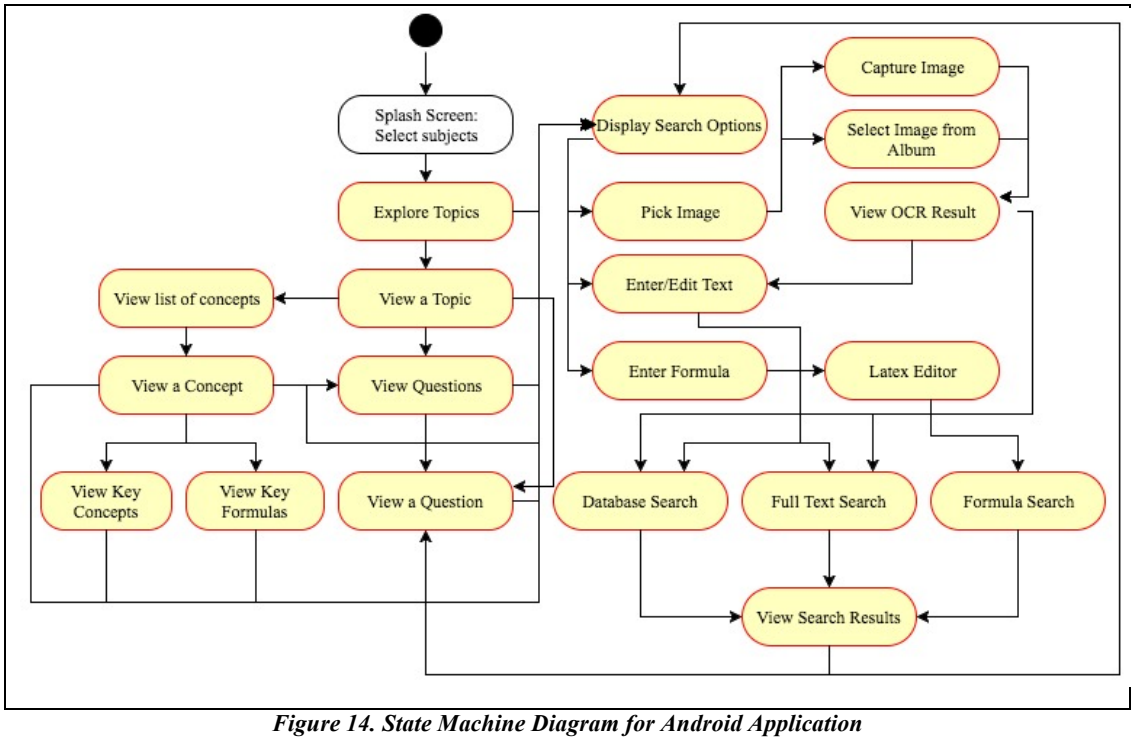


Figure 14. State Machine Diagram for Android Application

4 WEB SERVER DEVELOPMENT

This chapter discusses about the web server implementation for MathQA which include the tools and libraries used, implementation details for each component and the app final features. Figure 15 depicts the server architecture that is used during the web server development.

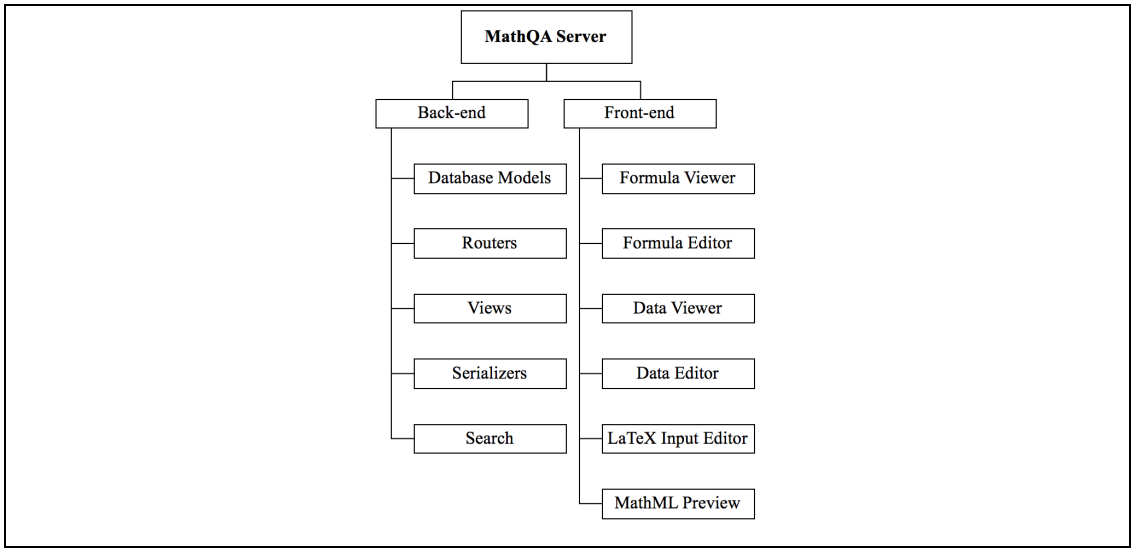


Figure 15. MathQA Server Architecture

4.1 Tools and Libraries

The back-end services for MathQA server are developed in Python using Django web framework, while the admin GUI is developed in HTML and JavaScript using AngularJS framework. Listed below are the related tools, frameworks and libraries that were utilised during web server development.

(1) MySQL, an open-source relational database system that runs on top of Django. All data inside the MathQA system are stored as collections of tables in MySQL.

(2) Django REST framework [23], a powerful and flexible Django toolkit for building Web APIs. It has easy installation and simplifies the building blocks that make up the REST Framework. This library is used to expose the REST API of the backend services available in the server.

(3) latex2mathml parser [24], an open-source library which generates MathML string from a LaTeX string.

(4) Django Haystack [25], a library that provides modular search for Django. It features a unified, familiar API that allows developers to plug in different search back ends without having to modify code implementation.
(5) Whoosh [26], is a fast, full-text indexing and searching library implemented in pure Python.
(6) Natural Language Toolkit (NLTK) [27], a platform for building Python applications that work with natural language data. This library is used to perform text pre-processing for Mathematical Document text retrieval.
(7) HTML, CSS and AngularJS Material [20], to build admin GUI for creating, editing and analysing data that contains LaTeX syntax.
(8) KaTeX [28], a fast, light-weighted LaTeX rendering engine built in JavaScript.

4.2 Backend Development

4.2.1 Models

A Django model is a single, definitive source of data information. Each model in Django maps to a single database table. A model contains fields and data behaviours that translate into a field in the database table. From these models, Django automatically generates a database-access API, which allows the creation, retrieval, update and deletion (CRUD) operations to be applied on the database using plain Python.

4.2.2 Serializers

Django's serialization framework provides a mechanism for translating Django object models (usually a queryset), into other formats such as JSON or XML. In this project, a serializer is created for each database model so that the server's response can be returned in JSON format to the clients.

4.2.3 Routers

Based on the URL patterns of the HTTP requests, the incoming request will get dispatched to the appropriate handler method. From Figure 16, the back-end services can be divided into two parts: database services and search services. For Database service, APIs for performing CRUD operations are supported. Public users only have read-only access to the database; therefore, retrieval and listing operations are provided. Database operation and formula APIs for developing formula-based retrieval may change the database, therefore these operations are only allowed for

authenticated admins. List of all the REST APIs is available in APPENDIX I AVAILABLE REST API

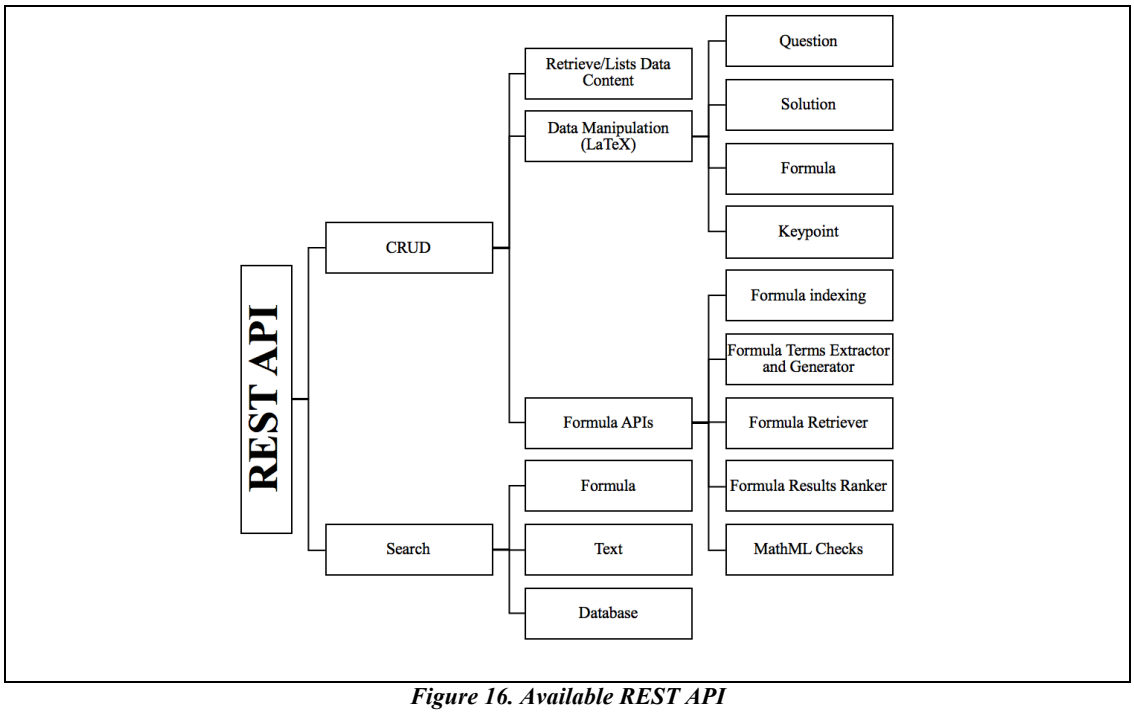


Figure 16. Available REST API

4.2.4 Views

Django views are request handlers. They are a set of Python's functions and classes that accept a request and return a response. During the implementation, both class-based and function-based views are used to handle a user request.

4.2.4.1 Class-Based Views

To achieve the goal of retrieving MathQA contents for the mobile app, read-only access REST APIs with Django models were created. These APIs were developed using class-based views which allow us to reuse common functionalities such as retrieving a list of objects and retrieving a specific object. Django query filters were also implemented to allow users to retrieve objects that satisfy a certain condition. Examples of this API request can be seen in Table 7. Note that the retrieved objects will be returned in JSON format.

Table 7. Read-Only Access Retrieval APIs

Retrieval Type	HTTP Requests	Actions
Listing objects	`GET /questions`	Retrieve all questions from the database
Retrieving an object	`GET /questions/1995102015`	Retrieve a question which id is 1996102015
Listing objects with query filters	`GET /questions/?concepts=1`	Retrieve all questions that are related to a concept with id of 1

4.2.4.2 Function-Based Views

These views are used to handle non-simple requests such as searching; database creation, update, delete (CUD) operations and formula indexing.

4.2.4.2.1 Search APIs

Search requests are a GET request to the server with the search type and query embedded inside the URL parameters. The implementation detail for these search APIs will be explained in chapter 5.

Table 8. Search APIs

Search Type	Example Requests	Actions
Database search (`type=d`)	`GET /search?type=d&query=curve%20has%20gradient%20$$e^{4x}%2Be^{-x}$$`	Perform database exact search for query *`curve has gradient $$e^{4x}+e^{-x}$$`*
Full text search (`type=t`)	`GET /search/?type=t&query=curve%20has%20gradient%20$$e^{4x}%2Be^{-x}$$`	Perform full-text search for query *`curve has gradient $$e^{4x}+e^{-x}$$`*
Formula Search (`type=f`)	`GET /search/?type=f&query=\sin%20x`	Perform formula search for the query *`\sin x`*

4.2.4.2.2 Create, Update, Delete (CUD) Operation APIs

CUD operations can only be executed by authenticated admins. These CUD operations are important especially for fixing and updating LaTeX contents.

4.2.4.2.3 Formula Indexing APIs

Formula indexing process must be performed prior to providing the formula retrieval service. All the existing formulas shall be reindexed. The indexing process starts with truncating the FormulaIndex table. Then, we will iterate through all formulas to generate the formula terms. These formula terms will be inserted into an entry in the FormulaIndex table. If a formula term does not exist yet in the table, a new entry will be created. Otherwise, then the current formula will be added to the entry's collection of formulas.

4.3 Front-end Development

The Admin Graphical User Interface (GUI) is developed for admins for creating LaTeX formulas; inspecting and manipulating LaTeX contents from the MathQA database and evaluating search performance. The building blocks of an AngularJS application can be decomposed into three parts:

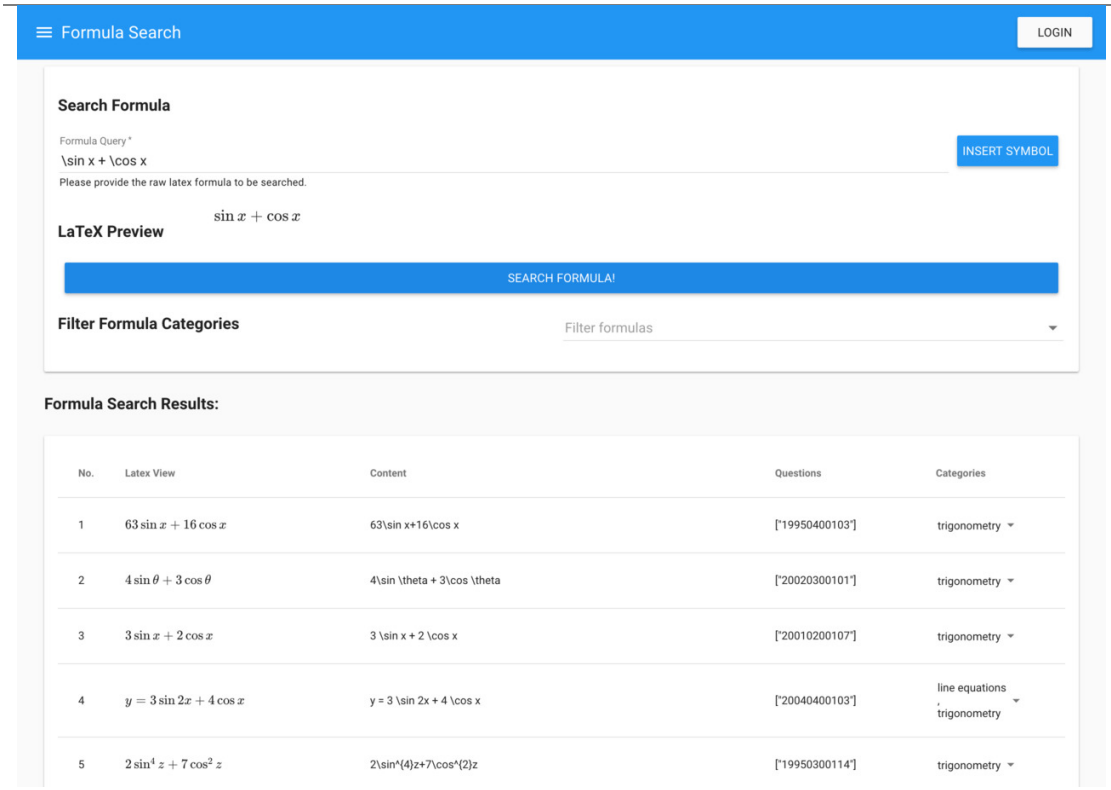
(1) Data services, which are responsible for retrieving data from the MathQA server.
(2) View controllers, which control the UI behaviours and the logic between UI components and Data services.
(3) HTML templates, which provide the UI components of the GUI.

From the requirements and state machine diagrams, admin GUI webpages were developed to help with analysing, fixing, and standardising LaTeX content of the data, performing experiments with mathematical document retrieval services, and evaluating the retrieval performance.

Table 9 demonstrates the features of the admin GUI.

Table 9. Admin GUI Webpages and Features

1) Formula Search Page



Features

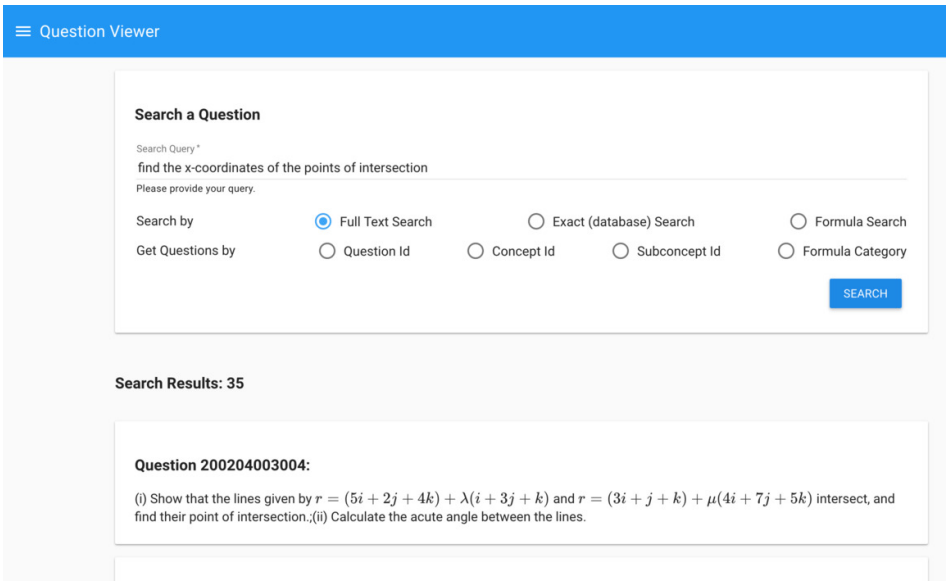
(1) In this page, admin can enter a formula query to perform formula search.

(2) The query can be entered manually using the editable text, or by utilising automatic LaTeX symbol generator through `INSERT SYMBOL` button.

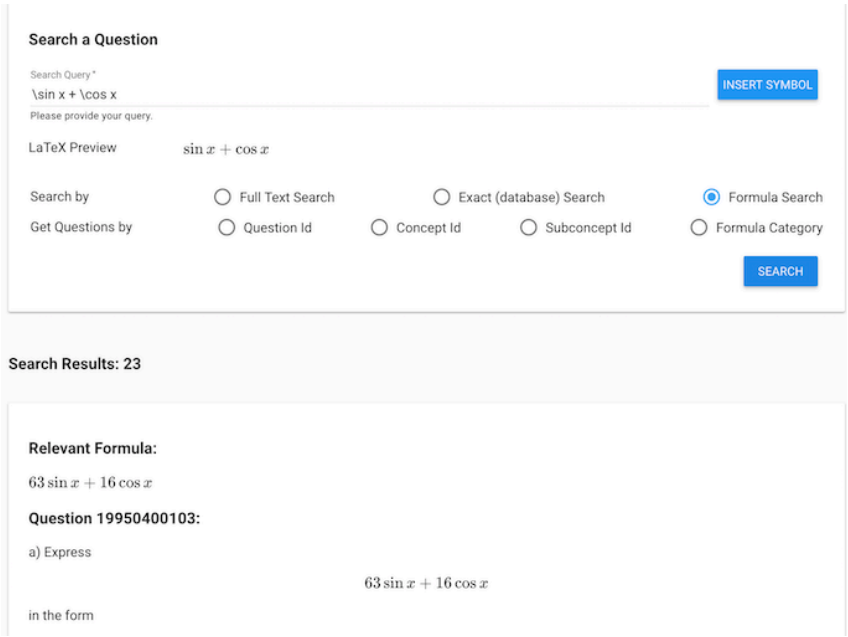
(3) The formula search results are returned as a row in the formula results table in decreasing order of relevance.

2) Question Search Page

General View



Formula Search



Features (1) In this page, admin can enter a query to perform maths question retrieval. There are two main methods of retrieving questions: using mathematical document retrieval techniques and Django queryset filters.

i. For mathematical document retrieval, a list of questions will be returned in descending order of relevance.

ii. For queryset filters, questions can be returned filtered based on the chosen filter type (i.e. if the user chose concept id, the list of questions under the same concept id will be returned).

(2) Special for formula search, these alterations are made:

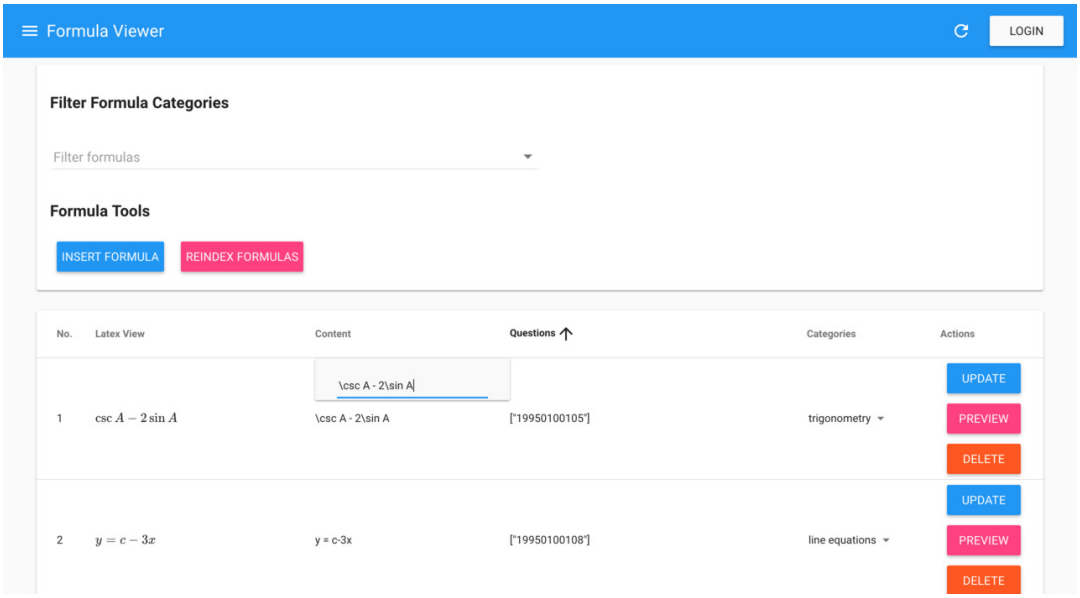
i. Input editor changed to cater for LaTeX input.

ii. The LaTeX preview and `INSERT SYMBOL` button are displayed.

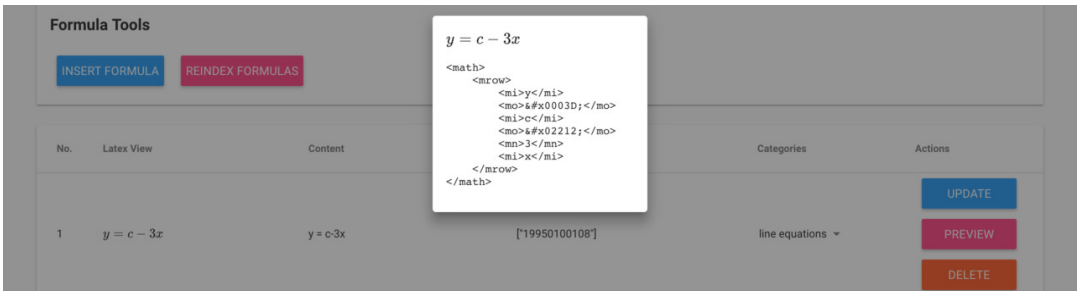
iii. The formula search results will contain relevant formula and the questions that contain the relevant formula

3) Formula Viewer Page

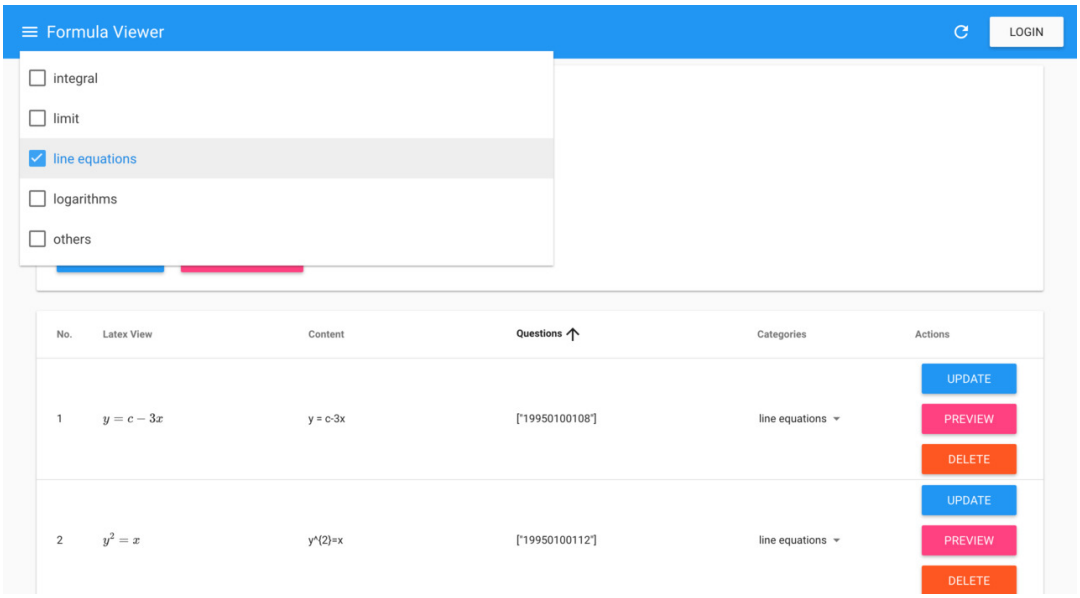
View All Formulas



MathML Preview



Filtered Formula (by Line Equations)



Features (1) In this page, admin can view the available formulas in the database. This page is important in order to inspect LaTeX syntax and performing a modification to fix errors or LaTeX syntax standardisation purpose.

(2) Each formula is represented as a row in the formula table. There are three actions available for each formula: update/ deletion which performs database update/ deletion on the edited formula and preview which display the MathML representation of the edited formula.

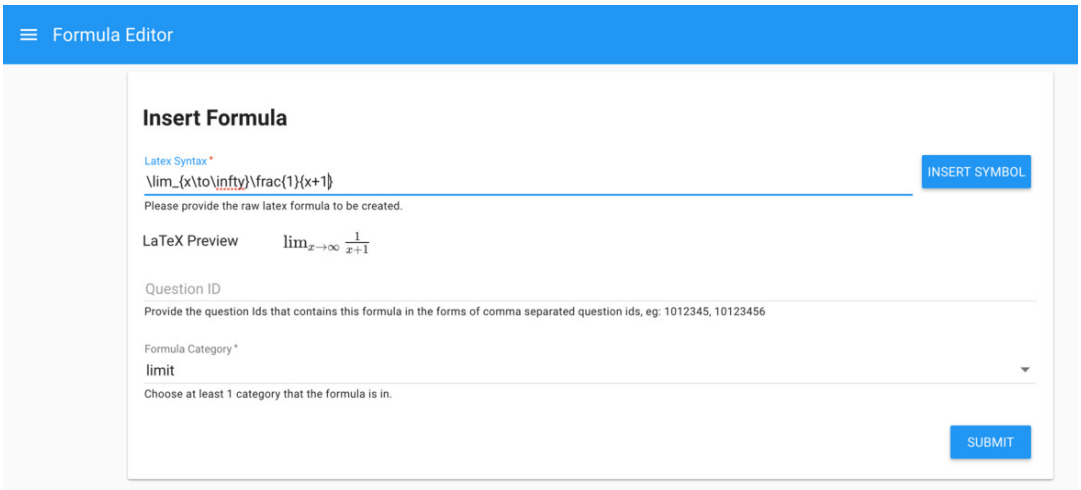
(3) Each field in the table is editable, therefore, the user can edit the formula content, questions, and formula categories respectively.

(4) A formula category filter is provided to filter formulas based on the selected categories.

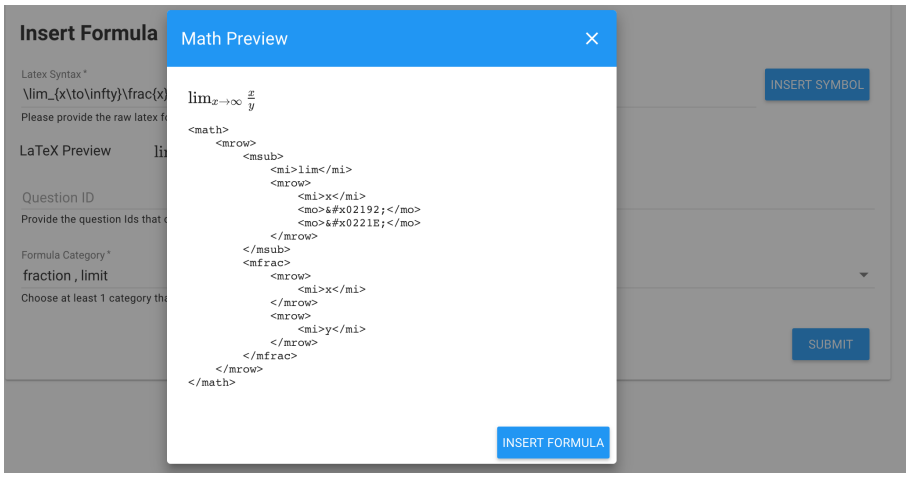
(5) `ReindexFormulas` button is provided to reindex all the existing formulas from the database. This is required because database modification needs to be reflected in the FormulaIndex table prior to retrieving relevant formulas.

4) Formula Editor

Insert Formula Form



LaTeX to MathML Preview



Features (1) This page is exercised by admin to insert a formula into the MathQA database.

(2) In this editor, user can enter the formula LaTeX through text input or via automatic LaTeX symbol generator.

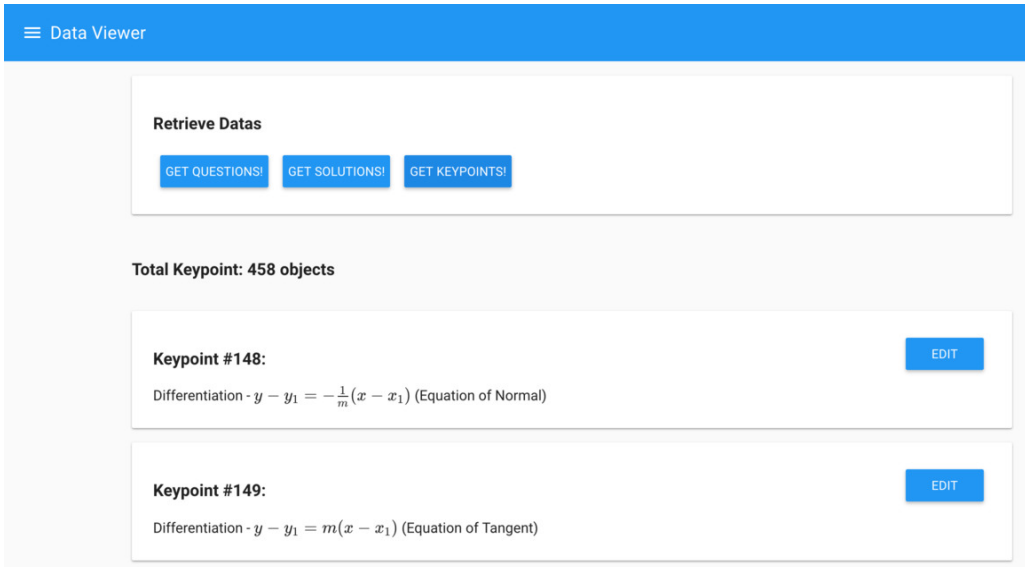
(3) User is required to provide the LaTeX input and formula categories for the formula. The user may also provide the question IDs where the formula is contained, but this step is not necessary.

(4) When a user submits the formula, a preview dialog will be popped up to provide admin with the information about the MathML representation of the LaTeX formula. If it looks correct, admin can proceed to insert

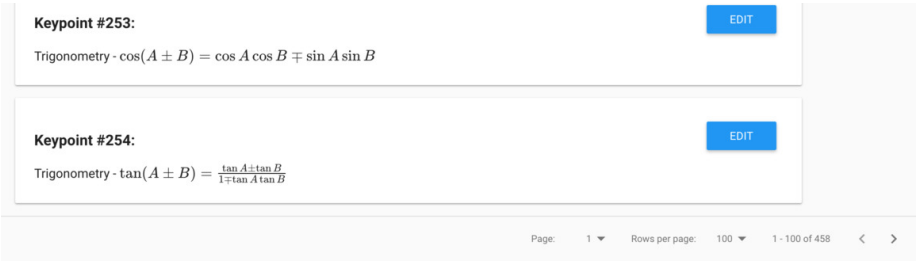
the formula into the database, otherwise, the formula insertion shall be cancelled.

5) Data Viewer

Main Display



Bottom Navigation



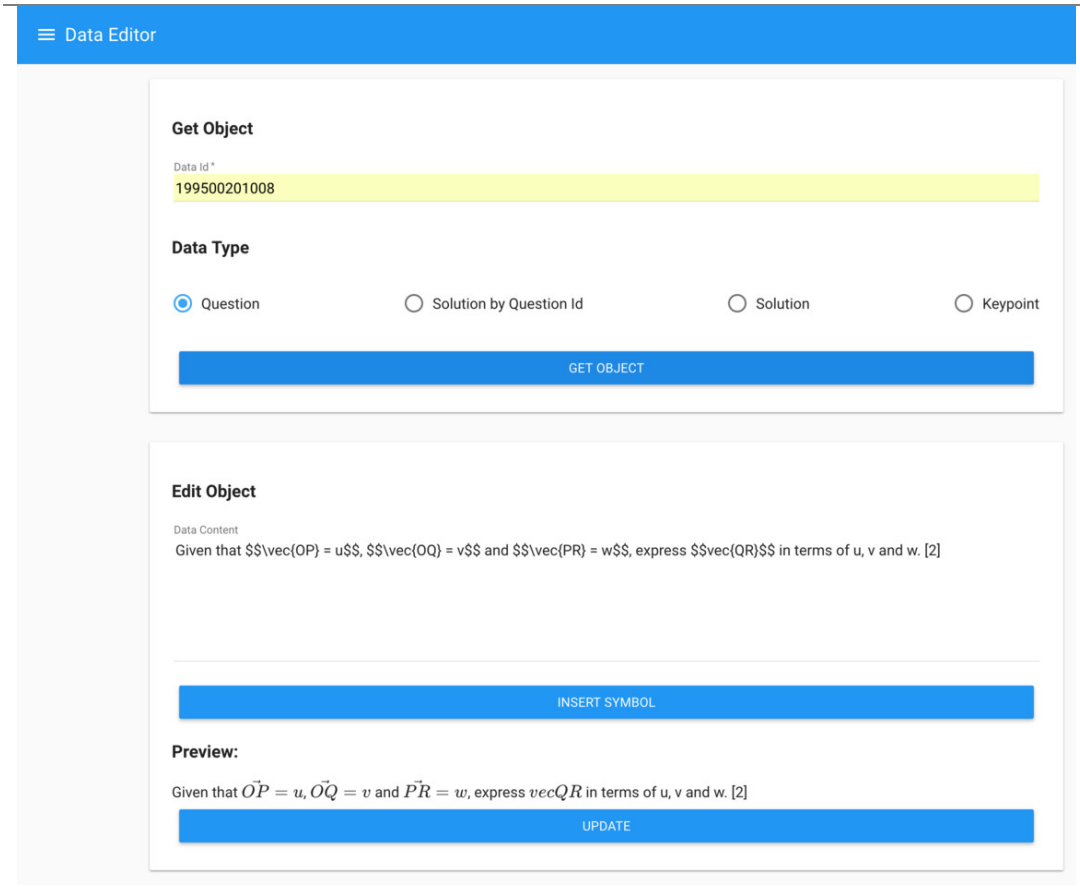
Features (1) This page provides a way for the admin to view the data from MathQA server that may contain LaTeX content. These data include questions, solutions and keypoints.

(2) When the user chose to retrieve data, all of these data will get displayed to the user.

(3) To avoid a slow loading time when the data size is too big, the list of data is paginated. Through pagination, 100 is set to be the maximum number of items that can be displayed on each page.

(4) At the bottom of the list, a page navigation is provided. Therefore, user can freely navigate between pages.

6) Data Editor

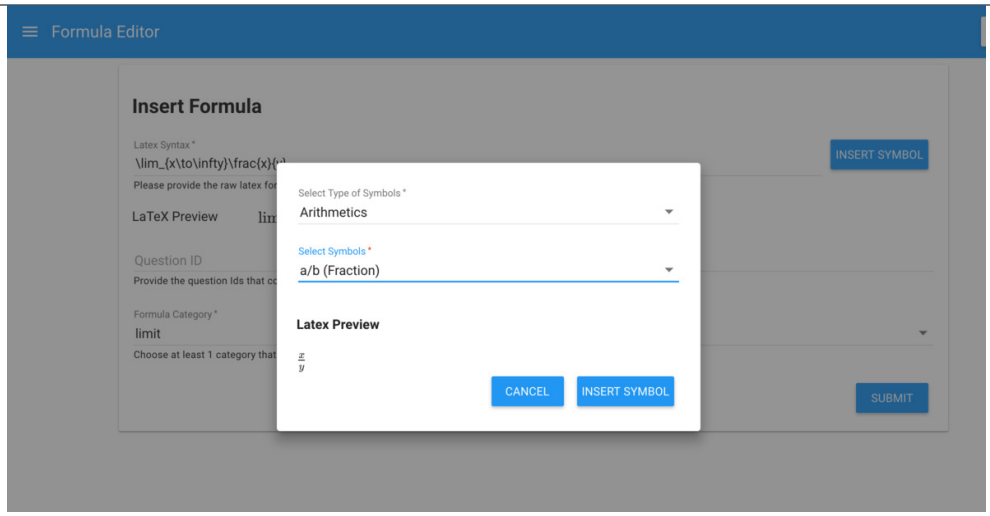


Features (1) At this page, admin can inspect the content of an object. This data editor is created to edit MathQA data that contain LaTeX syntax.

(2) An object is retrieved from server's REST API through its object ID.

(3) Admin is able to edit LaTeX content through input editor or LaTeX symbol generator.

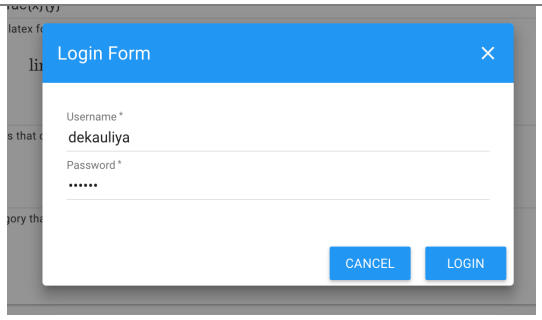
7) LaTeX Symbol Generator Dialog



Features (1) This dialog editor is used for inserting LaTeX. It is developed to ensure that the LaTeX content that is going to be inserted / modified to the database will always follow the syntax that has been tested and proven to be parsable into MathML. This is important since MathML representation is crucial during mathematical document retrieval development.

(2) This dialog is shown when the `INSERT SYMBOL` button is clicked.

8) Login Authentication



Features Admin needs to be authenticated in order to perform database manipulation such as create / update / delete operations and formula reindexing.

4.4 Discussions

During the web server development, the author has achieved the following objectives:

(1) Designed and developed a web-based system that provides educational mathematical contents and mathematical document retrieval services. From each

available service, the author has developed the respective REST APIs to be consumed by Android.

(2) Developed a robust mathematical document retrieval service composed of text-based and formula-based retrieval techniques.

(3) Developed an admin GUI that can help inspecting and correcting LaTeX syntax in the mathematical documents. The author hoped that this tool can be used by future students to improve the current mathematical document retrieval system.

(4) Incorporated good software engineering principles through the application of class based views and producing maintainable code through organisations of modules.

5 MATHEMATICAL DOCUMENT RETRIEVAL

This chapter discusses the implementation details of the retrieval services available in the back-end server. The retrieval service for mathematical document retrieval can be divided into two parts, text-based retrieval and formula-based retrieval. Experiments conducted to evaluate both retrieval services were also discussed and analysed in this chapter.

5.1 Text-Based Retrieval

Two types of text-based retrieval were developed: database search and full-text search. The implementation details of these retrievals are explained as follows.

5.1.1 Database Search

This search is implemented using Django query set `contains` filter. In this type of retrieval, there is no document or query pre-processing. The expected results of this query are questions that have an exact match with the database content. The database search that is implemented is equivalent to the following SQL `LIKE` query.

Table 10. Exact Database Search Queries

Django Queryset	`questions = Question.objects.filter( content__icontains=search_query)`
SQL Statement Equivalence	`SELECT * FROM Question WHERE content ILIKE '%[search_query]%'`

The operation of this search is similar to performing a simple intersection set operation between query terms and question terms in the database with an additional constraint: order of the term appearance matters. Django will go through each question and find the best match with the search query. Note the letter i in icontains is used to allow case-insensitive match with the database content.

Django queryset API can also be used to retrieve questions through advanced Django query filters. The list of all available filters is available in APPENDIX I AVAILABLE REST API section AI.2.

5.1.2 Full-Text Search

There are three main steps required for performing full-text search. The first step is the document pre-processing. Here, the math question contents in the database were

pre-processed to remove unnecessary characters with the help of NLTK library. Next, after the question data was pre-processed, an indexing table for Question content is built using Haystack and Whoosh. Through these two libraries, the Question Index table can be generated and the related questions can be found in decreasing order of query similarity score from HayStack SearchAPIs.

5.1.2.1 Document Pre-processing

This pre-processing stage aims to pre-process mathematical question content so that the end-result only contains meaningful and normalised word terms from the original question content. To perform question content pre-processing, the following actions are executed:

(1) LaTeX syntax removal from question content

This step is required because LaTeX syntax usually contains complex mathematical symbols and does not have meaningful words. Therefore, a simple pattern matching with the latex syntax is applied to remove latex from question content.

```
DOUBLE_DOLLAR_NOTATION = re.compile(r'\$\$([^\$]+)\$\$')
PAREN_NOTATION = re.compile(r'\\\(([^\(]+)\\\)')
BRACKET_NOTATION = re.compile(r'\\\[([^\[]+)\\\]')
def clean_latex(text):
    text = re.sub(fe.DOUBLE_DOLLAR_NOTATION, " ", text)
    text = re.sub(fe.PAREN_NOTATION, " ", text)
    text = re.sub(fe.BRACKET_NOTATION, " ", text)
    return text
```

Figure 17. Cleaning LaTeX from Question Content

(2) Non-alphabetical characters removal. After this step, only letters are preserved.

(3) Case-folding: transform words into their lowercase form.

(4) Non-meaningful word removal. Non-meaningful words are defined to be stopwords and words having less than 2 characters. Stopwords are words that do not contain important significance to be used in search queries. These non-meaningful words are filtered out from search queries because they give a vast amount of unnecessary information.

(5) Non-English word removal. At this step, words that do not exist in the English dictionary are filtered out.

(6) Text normalisation. At the last step, words are normalised into their root form (stem) using a Snowball stemmer.

Figure 18 shows an example of the cleaned text content after the processing stage. First, LaTeX syntax, non-letter characters and non-meaningful are removed. Then, the remaining words (e.g.: 'solve' and 'simultaneously') are transformed into their root form (e.g.: 'solv' and 'simultan').

```
"content": "Solve the simultaneous equations 2x - 4y = 13, 3x - 5y = $$\\frac{16}{2}$$. [3]",
"content_cleaned_text": "solv simultan",
```

Figure 18. Example of Full Question Content and Pre-processed Question Content

5.1.2.2 Term Indexing

At this step, Django Haystack is used to generate an inverted index of Question based on the word terms contained in the pre-processed cleaned question text field.

5.1.2.3 Querying

When there is a search query request, the query is first pre-processed using the same pre-processing method in section 5.1.2.1. Then, SearchQuerySet API from Haystack will be used to perform full-text search using the pre-processed query.

5.2 Formula-based Retrieval

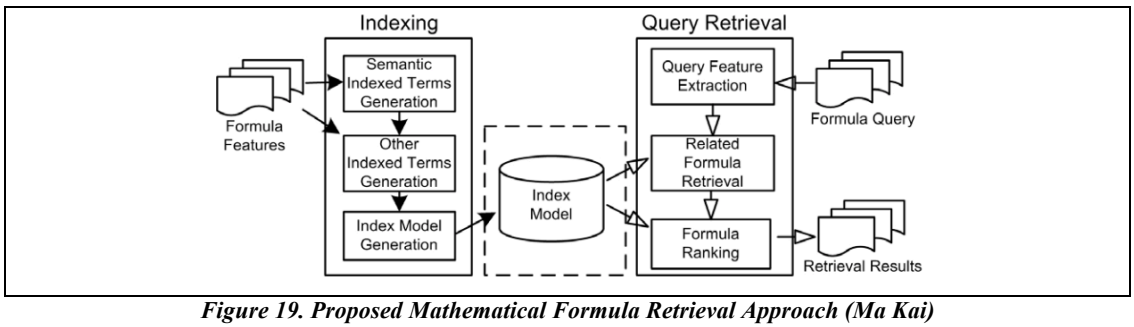


Figure 19. Proposed Mathematical Formula Retrieval Approach (Ma Kai)

To integrate the formula search to the system, the formula-based retrieval proposed by Ma Kai [1] was studied and implemented in pure Python. Based on the algorithm proposed in chapter 2.3.2, the implementation of the four main processes will be explained in detail.

5.2.1 Formula Feature Extraction

In this step, five formula feature terms (*in-order, ordered, structural, constant,* and *variable terms*) are generated from a LaTeX formula. First, latex2mathml library [29] is used to parse a LaTeX string into MathML tree string. Next, a DOM tree based on the MathML string is generated using the XML tree string parser available in Python. Building tree-like structure representation of the LaTeX string allows us to capture the semantic and structural relationship between the formula terms. Therefore, by going

through each node in the MathML DOM tree hierarchically, the five formula features can be extracted. All the supported formula terms for this feature extraction is available in APPENDIX II FORMULA TERMS.

Initially, a lot of MathML parsing problems were encountered due to LaTeX syntax errors and incompatible LaTeX syntax. Another problem was caused by the presence of the less than operator, < in the LaTeX formula. This error occurred because the < symbol can also be inferred as the opening tag for XML. Therefore, the parser will treat it as the opening tag instead of a mathematical operator. In order to solve this problem, the LaTeX string is first escaped into HTML entities prior to MathML parsing. This HTML escape is also useful for escaping the UTF-8 encoded characters, which also need to be handled in Python 2 because strings in Python 2 are bytes. Therefore, a proper encoding must be set.

The last problem was less obvious but has a significant impact on the retrieval performance. Although LaTeX representation of a formula is correct, the author found that some formulas were not able to be parsed correctly into a MathML string by the MathML parser. Different LaTeX representations of the same formula must be experimented to find the correct MathML representation of LaTeX.

In the middle of the project, an admin GUI was developed to help analyse LaTeX syntax and fix the errors or incompatibilities that exist in the mathematical document data. The admin GUI is useful for debugging formula-based retrieval because it allows admins to view the rendered LaTeX and view the MathML representation of a LaTeX string.

5.2.2 Formula Indexer

This process involves the creation of an inverted index formula table, which generates a table that maps a formula term to list of formula objects that contains that formula term. The list of properties of Formula and FormulaIndex table was described in Table 11. Note that many-to-many relationship is established between question and formula, therefore, a question may contain many formulas, and a formula can be related to many questions.

Table 11. Formula and FormulaIndex Table Model

Formula Table	FormulaIndex Table
content: LaTeX string of the formula *status*: boolean indicating whether or not a formula is indexed. *inorder_term*: inorder semantic terms of the formula *sorted_term*: sorted semantic terms of the formula *structure_term*: structural terms of the formula *constant_term*: constant terms of the formula *variable_term*: variable terms of the formula *questions*: list of questions that contain the latex formula *formula_categories:* the formula categories of the formula	*term_index*: the formula term *formulas*: list of formula objects that contains the index term *df*: number of formulas that contains the index term

As the MathQA server does not have formula data initially, author selected 508 LaTeX formulas to be inserted to the database. This data was inserted using Admin GUI manually to ensure the correctness of the LaTeX syntax and MathML representation. After the formulas were available, they are ready to be reindexed. During the reindexing process, the following actions are executed. First, the FormulaIndex table is truncated. Next, we will go through each formula in the database to extract the formula terms from it and update the FormulaIndex table entries with each generated formula term. If a formula term does not exist yet in the FormulaIndex table, a new entry in the FormulaIndex will be created and the formula containing that term gets inserted into the new entry's list of formulas. Otherwise, the list of formulas and document frequency of the existing entry will get updated.

5.2.3 Formula Retriever

Formula retriever receives a formula query from a client request. First, the query is cleaned from LaTeX delimiters. Next, formula features will be extracted from the formula query using algorithm 2.3.2.3, and the related formulas will be retrieved. The related formulas are obtained by performing a set union operation between query and formula terms. Having too many related formulas in the results could lower the retriever efficiency, therefore, a maximum candidate limit K is set. For this project, K is set to be 20. Once the number of candidates reached 20 relevant formulas, the retriever will stop looking for other formulas.

5.2.4 Formula Ranker

After relevant formulas are obtained, the similarity scores between these relevant formulas and the formula query is computed using step 2.3.2.4. In the implementation, the IDF values of the relevant formulas were first computed. These IDF values are used to compute the similarity scores between formula documents and formula query. During the ranking process, author modified the algorithm a bit to assign more weights to some of the formula operators such as integral, square root, fraction and series. The reason for this weight increase is to improve the formula retrieval accuracy. The full list of these formula functions and operators are available in APPENDIX AII.1 and AII.2 respectively.

The relevant formulas are then sorted in descending order of the query matching score. Since these formulas have reference to the question's id, the questions which contain the relevant formula can be retrieved and returned together with the relevant formula results.

5.3 Evaluation and Analysis

In this section, experiments were conducted to evaluate the performance of the available mathematical document retrieval services. This experiment is divided into two parts, Text-Based and Formula-Based Evaluation.

5.3.1 Text-Based Retrieval Evaluation

This evaluation aims to compare and analyse the search results produced from database and full-text retrieval.

5.3.1.1 Test Data Preparation

In order to perform the testing, several question contents were selected from the database with a differing characteristic:

(1) A query that has exact word-by-word match in the database.
(2) A query that has some terms available in the database, but all the words do not appear in together sequentially.
(3) A formula latex syntax query
(4) A formula latex syntax query that produces an English word after the extra symbols are removed.
(5) A query with some misspelt words.
(6) A query with all the meaningful terms misspelt.
(7) A full-text content of the question.

5.3.1.2 Tools

Text-based search APIs provided in section 4.2.4.2.1, the admin GUI from section 4.3 and Django debug toolbar were used as the instruments used for evaluating the text-based retrieval.

5.3.1.3 Evaluation Methods

As this section aims to compare and contrast between the database search and full-text search retrieval, a simple search result count and query processing time will be used for comparison.

5.3.1.4 Results and Analysis

Table 12 demonstrate the experimental results and performance of various search queries for text-based retrieval.

Table 12. Text-based Retrieval Results and Analysis

	Database Search		**Full-text Search**	
	# Results	**Time (ms)**	**# Results**	**Time (ms)**
1)	**Query with Exact Word-by-word Match in the Database**			
Text Query	`find the rate of change`			
Results	7	197.4	15	1458.63
Analysis	There are 7 question results in the database which contain exactly the same characters and order as the query. More results can be retrieved with the full-text search because it has a relaxed condition			

than database search where the matching terms in the results do not have to have the same exact order as the original query.

2) Query with Words that Do Not Appear Together Sequentially in the Database				
Text Query	`gradient of curve`			
Results	0	68.97	35	1738.06
Analysis	The exact words `gradient of curve` is not available in any question content of the database; however, the word `gradient` and `curve` exist in the database, therefore here full-text search is able to retrieve questions that contains the word gradient and curve.			

3) Formula LaTeX Syntax Query				
Text Query	`\int`			
Results	61	473.58	0	728.13
Analysis	By default, \int is removed because it is a LaTeX syntax. However, if it is not included in between the LaTeX delimiters, the following actions will happen: at pre-processing stage of formula search, \ will be removed. `int` has no meaning in English. Therefore, `int` will get omitted during the search and the full-text search results will return an empty result.			

4) Formula LaTeX Syntax that Contains English Words				
Text Query	`\sin x`			
Results	41	320.2	7	2000.959
Analysis	By default, \sin should already be removed from the pre-processing stage because \sin x is a LaTeX syntax. However, there might be some outliers where \sin x does not appear in between the LaTeX delimiters, therefore the full-text search is still able to retrieve questions. When this happens, the '\' will be first removed at the pre-processing stage. Then, since `sin` is an English word, the full-text search will search for question contents that contain the word sin.			

5) Query with some Misspelt Words				
Text Query	`A bank has an account for invstors for me`			
Results	0	38.95	2	1292.48

Analysis	Although the query contains a misspelling (invstors) or non-existent words (me) in the search query, the full-text based search is able to retrieve questions when the other terms of the query exist in the database.

6) Query with All Meaningful Terms Misspelt				
Text Query	`A bnk has an acount for invstors for me`			
Results	0	64.62	0	1287.617
Analysis	When the query has no meaningful terms, both full-text search and database search returns empty result.			

7) Full Question Text-content Query				
Text Query	`A bag contains 3 red balls, 2 white balls and 1 blue ball. Two balls are taken from the bag at random, without replacement. Find the probability that a) both balls are red, [1] b) the two balls are the same color, [2] c) the two balls are different colors.`			
Results	0	39.46	1	1725.056
Analysis	In this case, the full content of the question is obtained from full-text based search and not database search. The exact database search was unable to retrieve the question because the full question content in the database may contain text characters that are not easily observed through human eyes such as new line, extra whitespaces, other hidden utf-8 characters.			

5.3.1.5 Discussions

From the results, it is observed that the query processing time for database search is significantly faster than full-text search due to the pre-processing performed on the full-text search. As for the returned results, database search generally generates fewer results when supplied with a text-only query. However, this does not necessarily mean that the full-text search always performs better than the database search. These two types of searches serve a different purpose.

The database search is best used to look up for an exact match between text query and the database content. This search outperforms full-text search when non-alphanumeric characters exist in the search query, and the query that contains these non-alphanumeric characters exist a lot in the database. Moreover, if the user is

knowledgeable about LaTeX syntax, the user can supply the LaTeX syntax directly as a query to find the exact LaTeX syntax match from the database. This is possible because the database search does not perform any pre-processing on the text data. Therefore, LaTeX syntax and non-alphanumeric symbols are not removed.

The full-text search is best used to find most relevant documents with the query that have a more relaxed constraint than database search. The query terms for full-text search can contain other characters (such as term suffixes in a string), and allow terms to appear in a different order appearance from the question content in the database. The pre-processing performed on both query and question document in full-text retrieval allows this retrieval to generate more results than a database search. This happens because words like singular, plural, verbs with suffixes are reduced into their original form during the text normalisation process, hence more relevant documents can be retrieved.

5.3.2 Formula Retrieval Evaluation

5.3.2.1 Test Data Preparation

To conduct the experiment, the author first observed the available formula patterns from the questions. From the observation, common patterns of the formula are extracted and created into 13 formula categories:

Table 13. Formula Categories

1	Absolute	5	Inequality	9	Logarithms	13	Trigonometry
2	Differentiation	6	Integral	10	Polynomial		
3	Exponential	7	Limit	11	Series		
4	Fraction	8	Line Equations	12	Surds		

Creation of formula category is important to ensure that the formula-based search evaluation can be evenly distributed and remove bias to a particular type of formula. Then, for each formula category, the author selected about 50 formulas from the available question contents to be inserted into the Formula table. Differentiation, and limit formulas were excluded from being created due to the scarceness of the data. These formulas were inserted manually to ensure that it has the correct syntax and correct MathML representation. After the selection, 508 formulas were created to evaluate the formula-based retrieval performance.

Based on the available formulas, the author then selected 5 formulas (2 formulas for series formula due to data limitation) from each formula category to be used for test queries. The total of 52 test queries was selected for testing. The list of these test queries is available in APPENDIX III FULL FORMULA SEARCH EVALUATION

5.3.2.2 Tools

The formula-based search APIs provided in chapter 4.2.4.2.1, and the admin GUI with KaTeX from chapter 4.3 were used as the instruments for evaluating the formula-based retrieval.

5.3.2.3 Evaluation Methods

To evaluate the formula retrieval algorithm, the author first performed a formula-based search using all the 52 test queries and extracted the top-10 formula results. These top-10 formula results are then given a relevance label of 0 and 1, with 1 being relevant and 0 being irrelevant. An example of how this is done can be seen in Table 15.

After all the query results are labelled, two common evaluation metrics were used to evaluate the document retrieval system. These metrics include Precision@K and the Mean Average Precision (MAP) @K with K = 5 and K = 10.

5.3.2.3.1 Precision at K (P@K)

Precision at k documents (P@k) is the fraction of the top-k retrieved items that are relevant to the query.

$$Precision = \frac{\#\text{relevant items}}{\#\text{retrieved items}}$$

Figure 20. Precision Formula

5.3.2.3.2 Mean Average Precision (MAP)

MAP is the mean of the average precision for the top k retrieved items of all the test queries. Since 52 test queries were used during the testing, the MAP computed will be the mean of the AP@K computed for all the 52 test queries.

$MAP = \frac{1}{\|Q\|}\sum_{i=1}^{\|Q\|}\frac{1}{m_i}\sum_{k=1}^{m_i} Precision(rank_{ik})$	Q: Test queries m_i: number of items retrieved for query i Precision($rank_{ik}$): ratio of the top k retrieved items that are relevant.

Figure 21. Mean Average Precision Formula

5.3.2.4 Results and Analysis

Based on the experimental results in Table 14, it is observed that the inverted indexed technique proposed by Ma Kai had a promising performance. Average P@ {10} of 91.4% and MAP@ {10} of 97.5% were achieved from the test queries. The full AP and MAP scores for all the 52 test queries are available under APPENDIX III FULL FORMULA SEARCH EVALUATION RESULTS.

Table 14. Overall Formula Search Evaluation Results

AP {5}	**AP {10}**	**MAP {5}**	**MAP {10}**
0.965	0.9139	0.9919	0.9749777724

From the formula results, it is observed that the formula retrieval technique was robust enough to return relevant formulas in the first top five results. Interesting results, however, can be observed with the example query of `x+1 < 7 < x+3` as demonstrated in Table 15, where irrelevant results (polynomials) were returned in the 6 last results. The author suspects this result is caused by the insufficient number of the formula data, where there are only four formulas having the form of [expression1] < [expression2] < [expression3] available. Furthermore, if the search results were observed, it can be seen that only positive polynomial expressions were returned. This is because, in the original query, there are two addition operators available, therefore the formula retriever will try to find a relevant formula that contains at least two addition operators.

Table 15. Formula Search Results for Query x+1 < 7 < x+3

No.	Latex View	Content	Relevance	AP
1	$x+1<7<x+3$	x+1 < 7 < x+3	1	1
2	$-8<2x+3<11$	-8 < 2x + 3 < 11	1	1
3	$-5<2x+3<1$	-5 < 2x + 3 < 1	1	1
4	$-4<\|1+x\|<3$	-4 < |1 + x| < 3	1	1
5	$2x^2+7x+9$	2x^{2}+7x+9	0	0.8
6	$14x^3+ax^2+bx+10$	14x^{3}+ax^{2}+bx+10	0	0.67
7	$2x^3+ax^2+bx+3$	2x^{3} + ax^{2} + bx +3	0	0.57
8	$2x^3+px^2-12x+q$	2x^{3}+px^{2}-12x+q	0	0.5
9	x^3-6x^2+ax+b	x^{3}-6x^{2}+ax+b	0	0.44
10	x^3+ax^2+bx-3	x^{3}+ax^{2}+bx-3	0	0.4
P{5}	0.8		AP{5}	1
P{10}	0.4		AP{10}	1

The full P@K and AP@K scores of all the test formulas are displayed in Figure 22 and Figure 23 respectively. Based on these results, it is shown that the proposed formula-based retrieval technique is stable among all types of formula and decent performance with the lowest P@K score of 82% and AP@K score of 92% among all types of formulas.

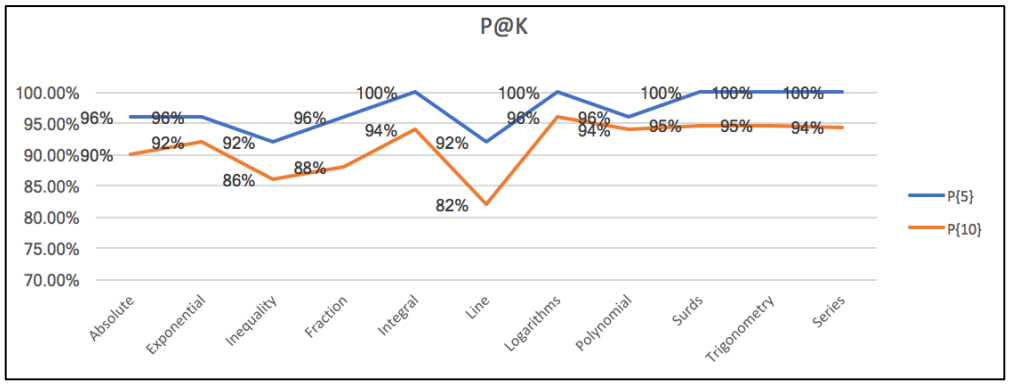


Figure 22. P@5 and P@10 Scores

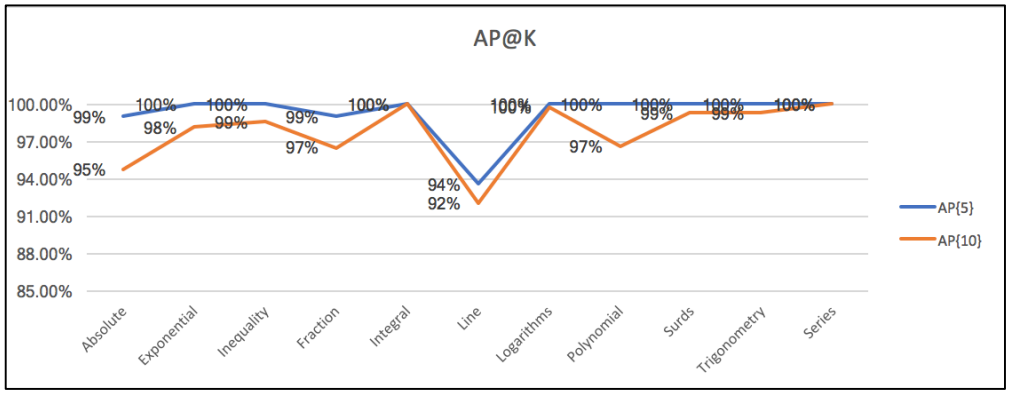


Figure 23. AP@5 and AP@10 Scores

5.3.2.5 Discussions

During the initial inspection of the data, the author observed that there were a lot of syntax errors and non-standardised formulas in the mathematical questions. A non-standardised formula is defined to be those formulas that have similar meaning but have different LaTeX representation. An example of this is \log x and \log (x) both of which represents the log of x, or x^2 and x^ {2} both of which represents the square of x. When the first attempt to evaluate the formula search performance was done (no LaTeX syntax standardisation), the formula retrieval accuracy was quite poor. This result occurs because when similar meaning formulas have different LaTeX representations, their parsed MathML representations may differ as illustrated in Table 16. Therefore, a LaTeX syntax standardisation step is necessary.

Table 16. Different MathML Representations of the Same log x Function due to LaTeX Syntax Difference

Latex Syntax	\log x	\log {x}	\log (x)
MathML Preview	$\log x$	$\log x$	$\log(x)$

	`<math>`
	`  <mrow>`
	`  <mi>log</mi>`
	`  <mi>x</mi>`
	`  </mrow>`
	`</math>`
		`<math>`
		`  <mrow>`
		`  <mi>log</mi>`
		`  <mrow>`
		`  <mi>x</mi>`
		`  </mrow>`
		`  </mrow>`
		`</math>`
			`<math>`
			`  <mrow>`
			`  <mi>log</mi>`
			`  <mrow>`
			`  <mo>&#x00028;</mo>`
			`  <mi>x</mi>`
			`  <mo>&#x00029;</mo>`
			`  </mrow>`
			`  </mrow>`
			`</math>`

However, inspecting and fixing LaTeX is not trivial without visual aid. Due to this reason, an admin GUI in chapter 4.3 was built and used to fix all the observed LaTeX syntax errors and standardising the similar meaning formulas. From author's observation with LaTeX syntax standardisations, author suggested the following recommendations in representing LaTeX formulas:

(1) Avoid the use of parentheses for simple formulas. This is because when we have a lot of function formulas that contain parentheses, two very different functions might be considered relevant due to the presence of parentheses in both LaTeX formulas. An example of this is log(x) and sin(x). During the experiments prior to LaTeX syntax standardisation, sin(x) formulas were found to be returned as the relevant formula results from the log(x) formula query and was even ranked higher than formulas that contain log x without the parentheses.

(2) Use curly brackets when having exponentials, square root, fractions and subsup operations. Although x^{2} and x^2 both represents the same formula, the different LaTeX representations may cause the parser to interpret them as two different formulas. Therefore, the use of {} is encouraged in power functions to also cater for complex exponent expressions.

(3) The order of subsup and supsub matters. These two operators are used to create superscript and subscript symbols. Due to the limitation of the parser, it was unable to parse the LaTeX formula into a correct DOM structure if sup operator is used before the sub operator. Therefore, to solve this problem author converted all the definite integrals to use sub operator before sup operator.

Table 17. Correct and Incorrect MathML Representations of Semantically Similar LaTeX Syntax.

Correct MathML with **\int_{0}^{3} x \mathrm{d}x**	**Incorrect MathML with** **\int}^{3}_{0} x \mathrm{d}x**
$\int_0^3 x\mathrm{d}x$	$\int_0^3 x\mathrm{d}x$

`<math>`
`  <mrow>`
`  <msubsup>`
`  <mo>&#x0222B;</mo>`
`  <mrow>`
`  <mn>0</mn>`
`  </mrow>`
`  <mrow>`
`  <mn>3</mn>`
`  </mrow>`
`  </msubsup>`
`  <mi>x</mi>`
`  <mi>mathrm</mi>`
`  <mrow>`
`  <mi>d</mi>`
`  </mrow>`
`  <mi>x</mi>`
`  </mrow>`
`</math>`
	`<math>`
	`  <mrow>`
	`  <msubsup>`
	`  <mo>&#x0222B;</mo>`
	`  <mrow>`
	`  <mn>0</mn>`
	`  </mrow>`
	`  <mi>x</mi>`
	`  </msubsup>`
	`  <mi>mathrm</mi>`
	`  <mrow>`
	`  <mi>d</mi>`
	`  </mrow>`
	`  <mi>x</mi>`
	`  <mrow>`
	`  <mn>3</mn>`
	`  </mrow>`
	`  </mrow>`
	`</math>`

6 ANDROID DEVELOPMENT

This chapter discusses the development of MathQA in Android which include the tools and libraries used, implementation details for each component and the app final features. Figure 24 depicts the android architecture that is used during the app development.

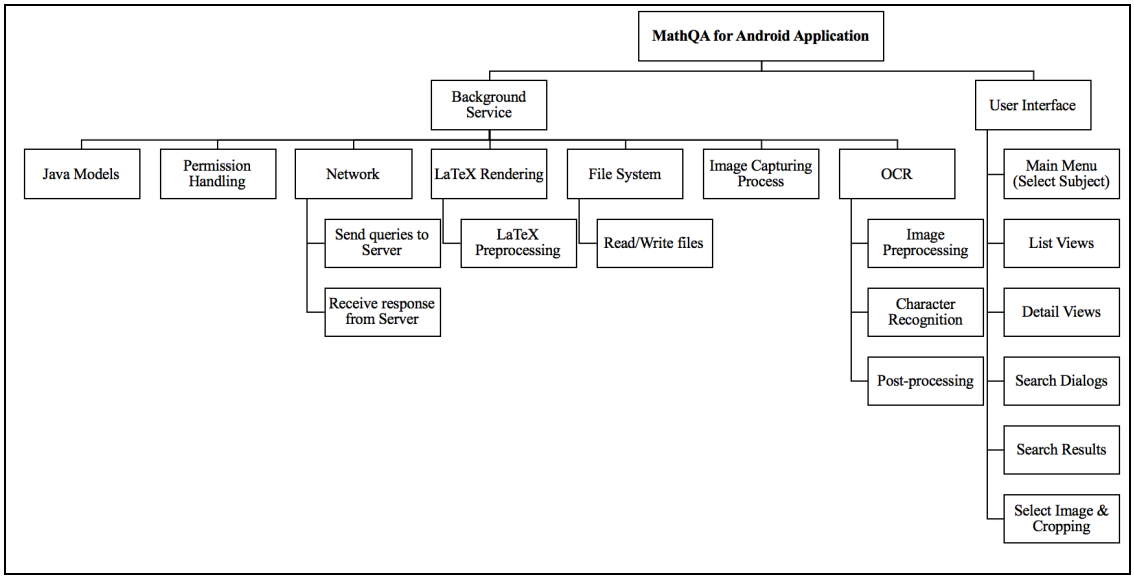


Figure 24. Android Architecture

6.1 Tools and Libraries

The whole development process of Android is developed in Java using Android Studio 2.3.

6.1.1 Application Framework

AndroidAnnotation framework was adopted during the Android app development. This framework offers practical features that aim to simplify the code implementation and produce maintainable code. In this project, this framework is used to:

(1) Apply dependency injections between Java classes, views, activities and fragments

(2) Bind views to its layout resource and events (e.g.: clicks or data events)

(3) Simplify the multi-threading processes between UI and background threads

(4) Communicate data or events between UI and background tasks

6.1.2 Material Design Views

To incorporate Material Design guideline during the app development, various Android views libraries were explored. From the experimentations, the following libraries are found to be suitable to be incorporated into MathQA app:

(1) MaterialValue library [30], is used to ensure that the resource values used for developing MathQA UI follow Material Design specification. These values include colour codes, different component sizes, and spacing.

(2) FlexibleAdapter library [31], which provides reusable recyclerview components. The library is extensible and offers predefined logic for different UI states. In the scope of this project, simple recycler ListView and ExpandableListView were utilised to display a list of items from MathQA server.

   i. ExpandableListView, a view that consist of a header item and subitems. Header item groups the subitems together and controls the display of subitems. When a header item is clicked, it toggles between the collapsing and expanding states to hide or show the subitems.

   ii. SimpleListView, a view which is the list of subitems with no header item.

(3) Floating Action Buttons represent primary actions of an application. They are distinguished by a circled icon floating above the UI and have special motion behaviours. For MathQA app, the different question search actions are promoted into Floating Action Buttons using the FloatingActionButton library [32].

(4) Dialogs are fragment which displays an overlay modal window within an activity that floats on top of the rest of the content. MaterialDialogs [33] library is used to provide customised material dialogs that prompt user's query for searching mathematical documents or displaying a progress dialog. This library is also used to develop a custom LaTeX input editor for inputting LaTeX formula query.

(5) Progress Activity [34], is utilised to provide users feedback regarding current app status such as loading, empty and error states.

6.1.3 Latex Rendering

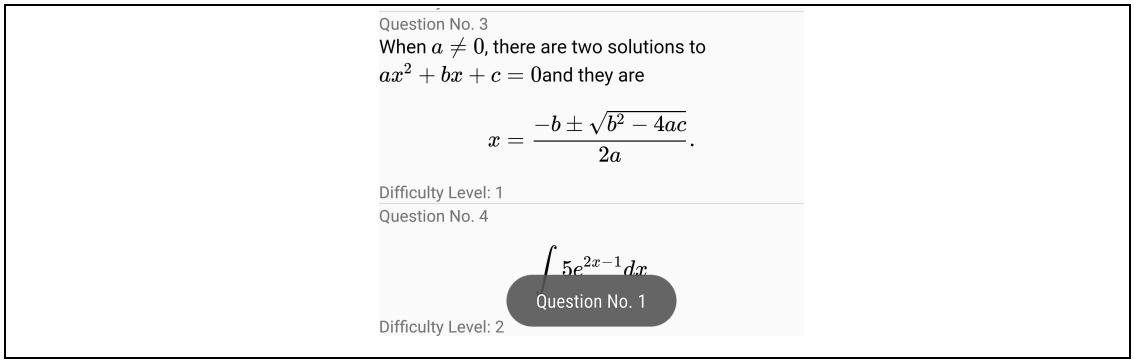


Figure 25. LaTeX Rendering

Native Android development does not support LaTeX rendering, moreover, the available Java LaTeX rendering engine requires JavaX or Swing, both of which incompatible with Android. Thus, a different approach was considered. The author found a library called MathView [35], which allows LaTeX rendering on Android through the use of WebView. WebView is a view widget in Android that allows the use of HTML and JavaScript. Therefore, JavaScript LaTeX rendering engine can be used to render LaTeX. This library supports two JavaScript LaTeX rendering engines: KaTeX and MathJax. After, comparing the two engines, KaTeX is then selected because it has a much faster rendering time and lighter weighted than MathJaX.

6.1.4 Network Access and REST API

(1) Retrofit library [36], is used to establish network connections and create HTTP clients for Android.

(2) RxJava 2 [37], is used to compose asynchronous and event-based network through observable sequences.

6.1.5 OCR Related Libraries

(1) ImageCropper [38] is a library that facilitates image cropping utilities.

(2) Image pre-processing. Document images are first pre-processed to achieve a better OCR performance.

i. Leptonica, an open-source library for performing image processing and analysis operations [18].

ii. Catalano Framework, an image analysis and pre-processing framework for scientific computing developed in Java and Android [19].

(3) Text Recognition

i. Tess-two, a library that provides Tesseract tools for Android [39].

ii. Google Text API, part of Mobile Vision API Framework developed by Google [20].

6.2 Background Services

6.2.1 Java Object Models

For each available object model in the MathQA server, a Java object model equivalent is created. These Java object models act as the container for storing data retrieved from the MathQA server.

6.2.2 Network Services

Network service was established between Android and the MathQA server using the retrofit library. Observable patterns were incorporated into the network service to allow network requests to be executed in parallel (non-blocking). The REST API methods for calling MathQA service and the appropriate plain Java object models for storing the server response were developed based on the available MathQA services. This service development is divided into two parts: Data Service and Search Service.

6.2.2.1 Data Service

Data service involves all the data retrieval requests to the server. All methods in DataService are translated into HTTP GET retrieval requests (see APPENDIX I). Based on the data type that the user requested, server's JSON object responses will get deserialized into a collection of the equivalent Java object models.

6.2.2.2 Search Service

Search service operates similarly to Data Service. However, in Data Service, the user did not supply any data during the request. In search service, both search type and search query parameter must be supplied by the user to perform mathematical document retrieval. There are three types of search supported:

Table 18. The Mathematical Document Search Services

Search Services	Actions
searchDatabase(query)	Perform a database search
searchText(query)	Perform a full-text search
searchFormula(query)	Perform a formula search

6.2.3 Displaying LaTeX Content

Before displaying the LaTeX content on MathView, an inspection is done on the LaTeX content to check if LaTeX delimiters are required. Alternative TextView was

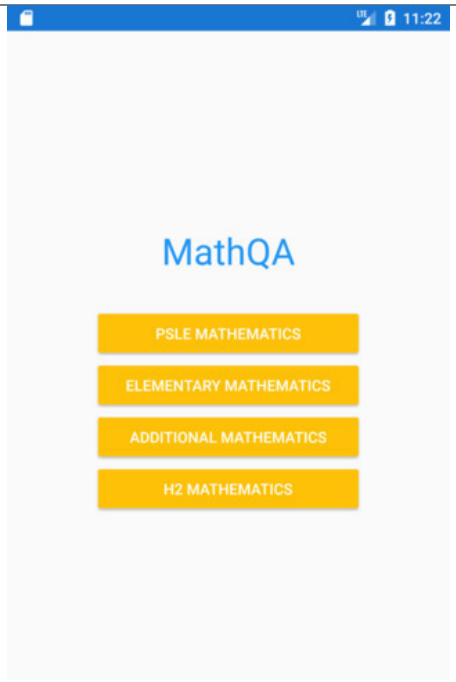
also provided to replace MathView when the LaTeX content is unavailable due to some errors with the LaTeX syntax.

6.3 Frontend

Based on the state machine diagram and the requirements elicited in the early stage of SDLC, the prototype is then developed into the following working Android app components.

6.3.1 Home Activity

Table 19. Home Screen Activity

User Interface	Activity
	*SelectSubjectActivity* (1) Offers four main subjects in the MathQA system to be selected.

6.3.2 Displaying Mathematical Contents from MathQA

Several views need to be implemented in order to display mathematical documents from MathQA server in an intuitive manner. To do this, commonly occurring view patterns were extracted and abstracted to allow reusable components as displayed in Figure 26. These reusable components include ViewPagerActivities, ListViewFragments, and DetailViewFragments.

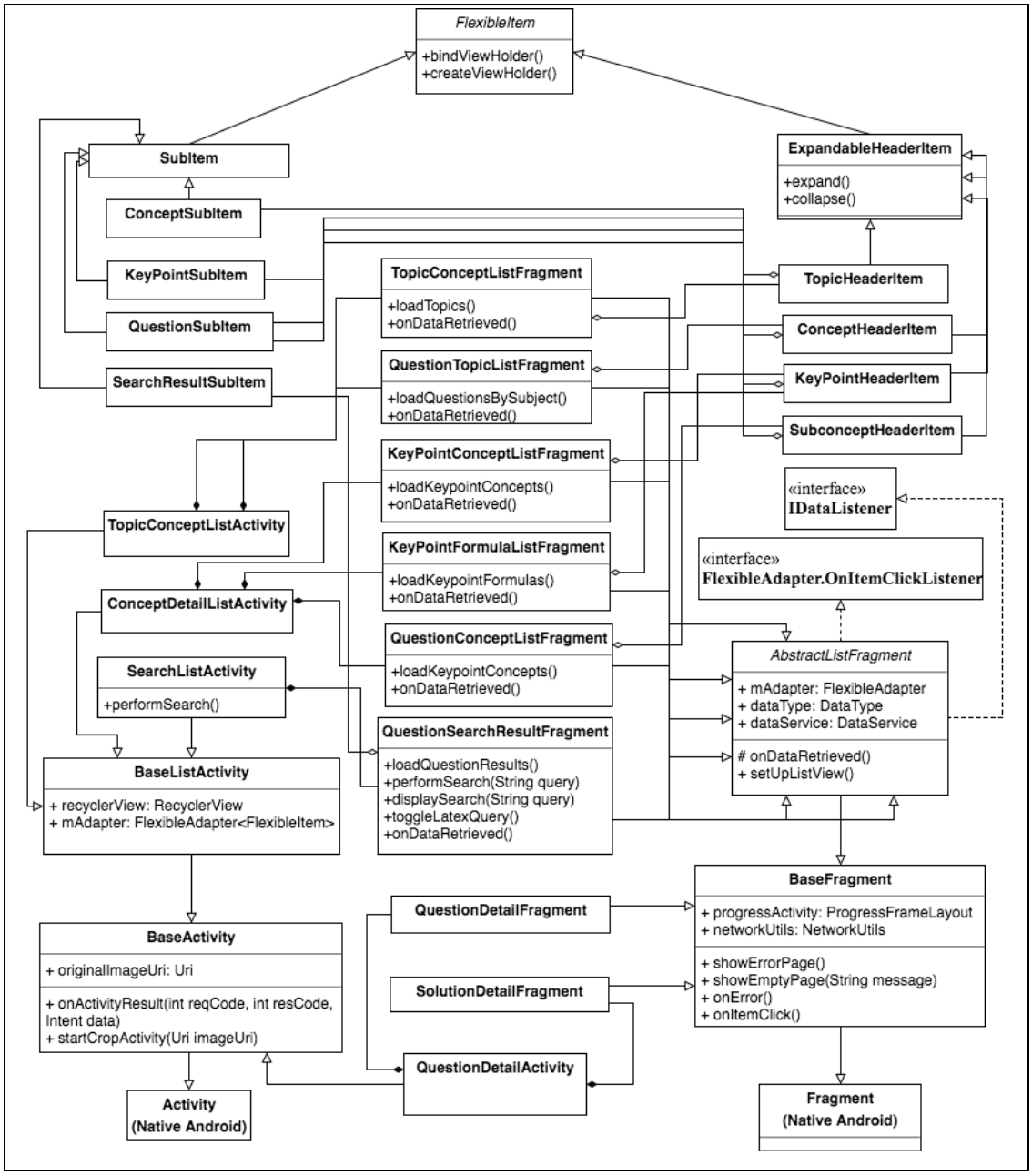
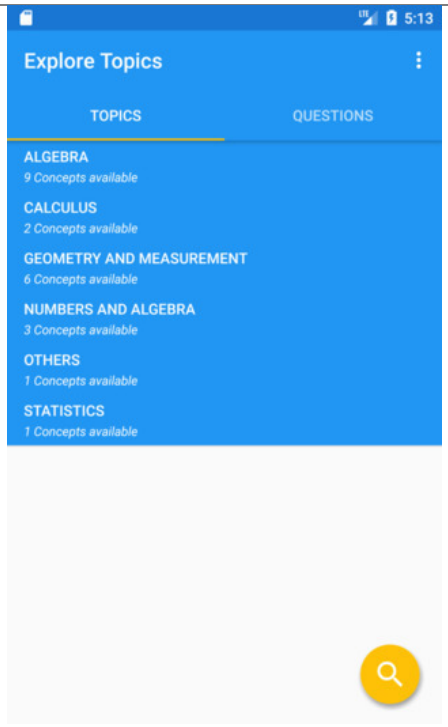
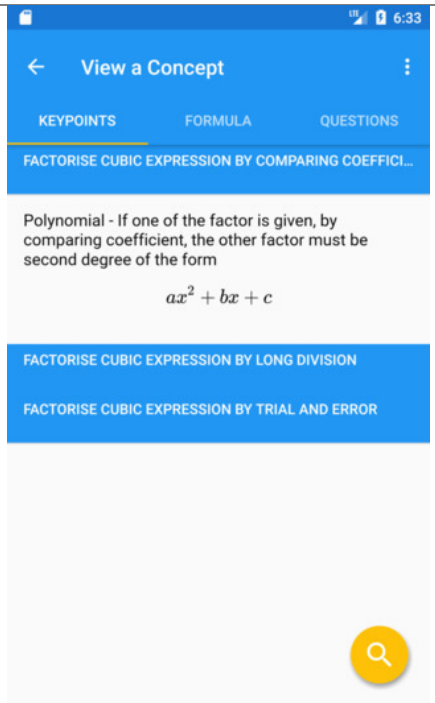
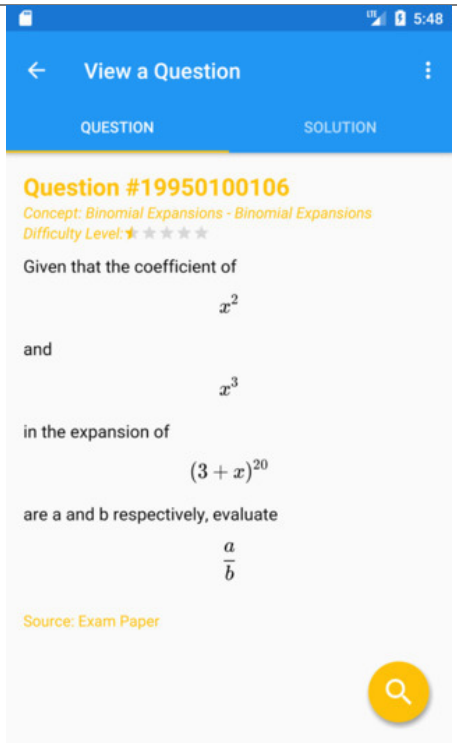


Figure 26. Class Diagram of Reusable View Components

6.3.2.1 *ViewPager Activities*

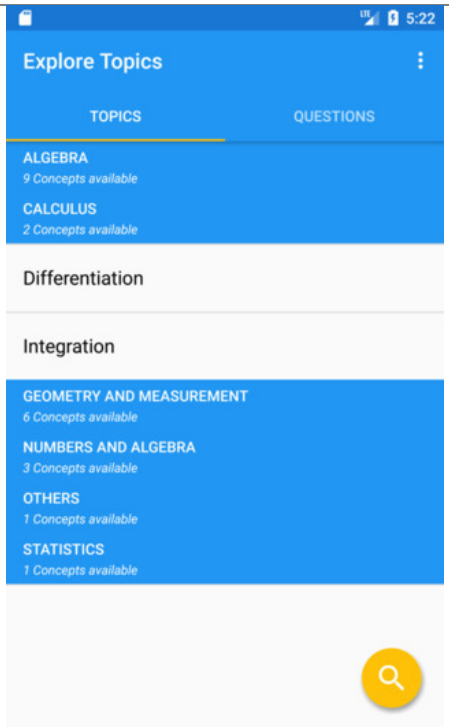
Table 20. ViewPager Activities

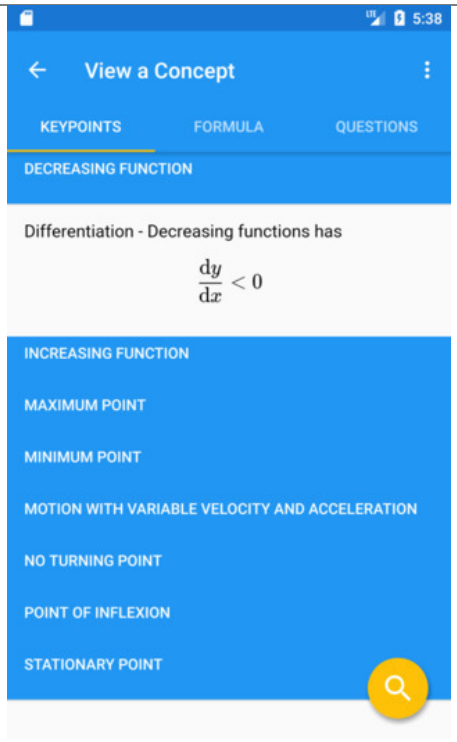
User Interfaces	Activities
	*TopicConceptActivity* (1) This activity will be shown after the user chose the subject. Therefore, the topics listed are available topics for a chosen subject. (2) This activity is a viewpager with two fragments: TopicConceptListFragment and QuestionConceptListFragment.
	*ConceptDetailActivity* (1) This activity will be shown when a concept is clicked from the TopicConceptListFragment (2) This activity is a viewpager with three fragments: KeypointConceptListFragment, KeypointFormulaListFragment, and QuestionConceptListFragment

	*QuestionDetailActivity* (1) This activity will be shown when a question item is clicked from any fragments and activities. (2) This activity is a viewpager which consist of two fragments: QuestionDetailFragment and SolutionDetailFragment

6.3.2.2 *ExpandableListView Activities*

Table 21. ExpandableListView Activities

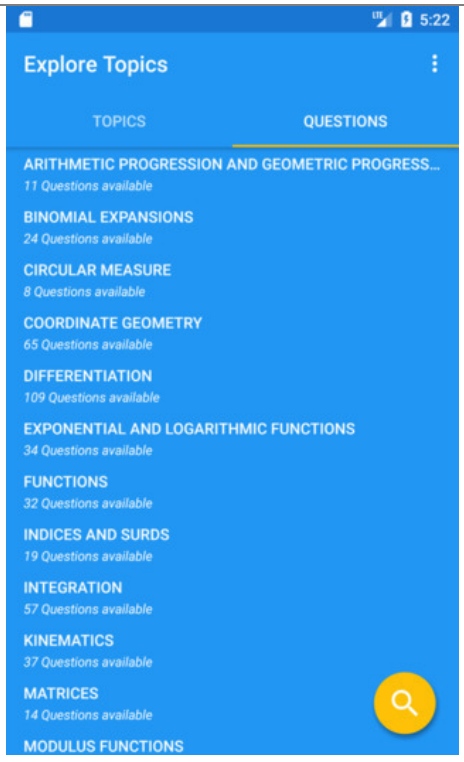
User Interfaces	Activity
	*TopicConceptListFragment* (1) This fragment is shown right after the user selected a subject from the home screen. (2) It displays topics as the headers which can be expanded or collapsed. (3) Concepts under the same topic are grouped under same topic header.



KeypointConceptListFragment and *KeypointFormulaListFragment*

(1) This fragment is shown after the user selected a maths concept.

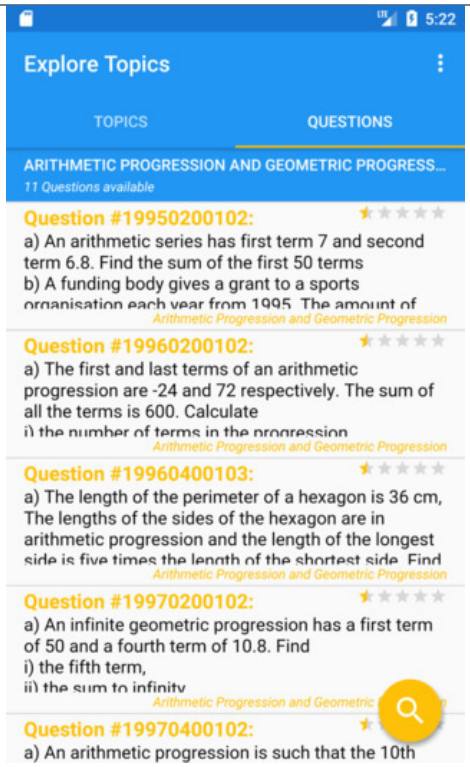
(2) The two fragments are quite similar as they both provide information about theories and formulas related to a maths concepts.



QuestionTopicListFragment and *QuestionConceptListFragment*

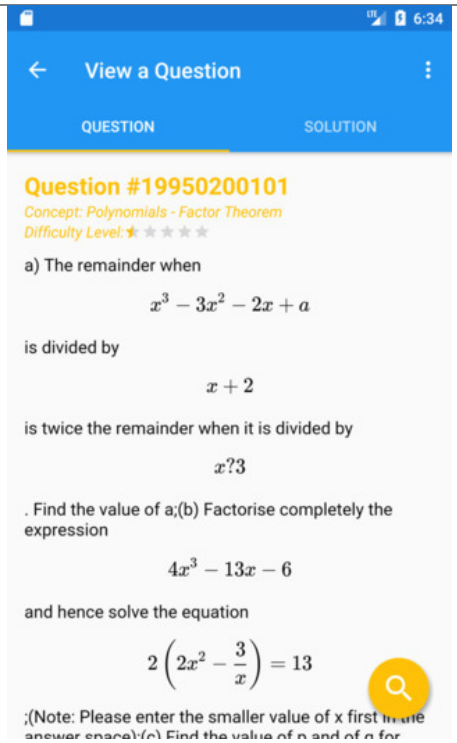
(1) These two fragments display a list of available questions with the level of detail that depends on the parent Activity.

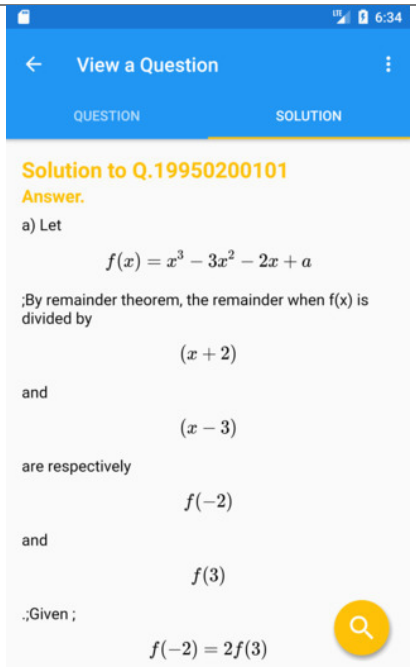
(2) For QuestionTopicListFragment, questions are grouped under the same concept, while in QuestionConceptListFragment, questions are grouped under the same subconcept.

	(3) Each question subitem in the ListViews is not fully displayed, but displayed in a preview form where users can see the question's key information and the first few lines of the question.

6.3.2.3 *DetailedView Fragments*

Table 22. DetailedView Fragments

User Interfaces	Fragments
	*QuestionDetailFragment* This fragment displays the full question content and question's metadata

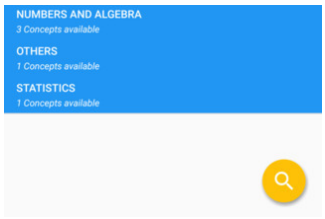
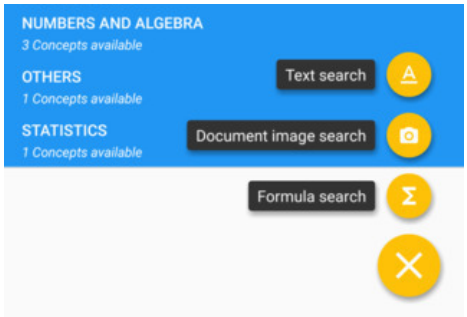
	*SolutionDetailFragment* This fragment displays the full solution content of a question

6.3.3 Search-related Views

For searching mathematical contents, different types of search input queries were developed. These input queries cater to the different types of input query methods, which include: text query, formula query, and captured text from a document image.

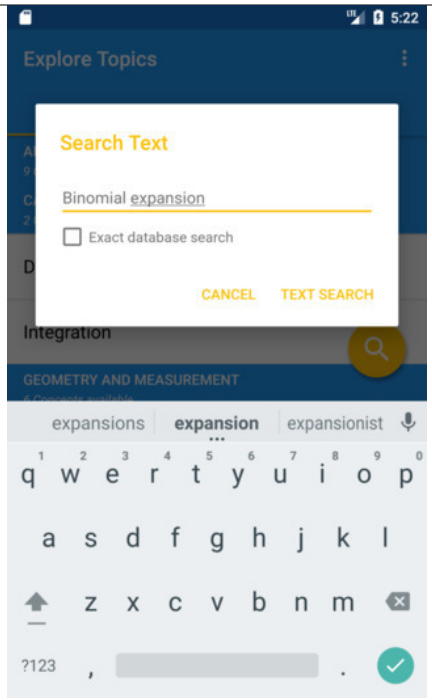
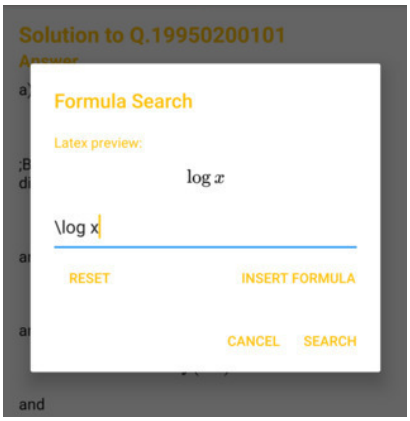
6.3.3.1 Floating Search Buttons (FAB)

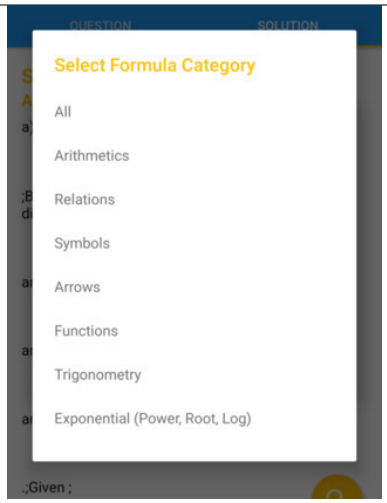
Table 23. Search Floating Action Buttons

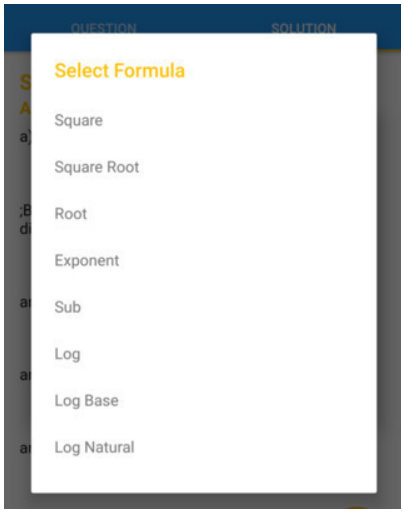
User Interfaces	Views
 	(1) There are 3 sources of input queries for performing a search in the MathQA app. Therefore, three FAB are created for each search text-based search, image-based search (through OCR), and formula search. (2) For text-based and formula inputs, a search input dialog will be displayed, while for image-based search, a new activity for capturing image will be created.

6.3.3.2 *Different Search Dialogs*

Table 24. Different Search Dialogs

User Interfaces	Views
	*Text Prompt Dialog* (1) This dialog is used to perform text-based search. (2) It consists of a text input where a user can type their query from keyboard. An option to perform a database search is also provided through a checkbox. (3) This dialog does not permit empty query. If that is the case, the search action button will be disabled.
	*Latex Editor Dialog for Formula search input query* (1) This dialog is used for formula-based search. (2) A LaTeX preview is provided in the dialog to show the user how the LaTeX string looks like after it is rendered. (3) Similar to the LaTeX symbol generator for Admin GUI, here a LaTeX symbol generator for generating LaTeX symbols in Android is developed. (4) The user has the option to edit the LaTeX formula through the text editor or by using the LaTeX symbol



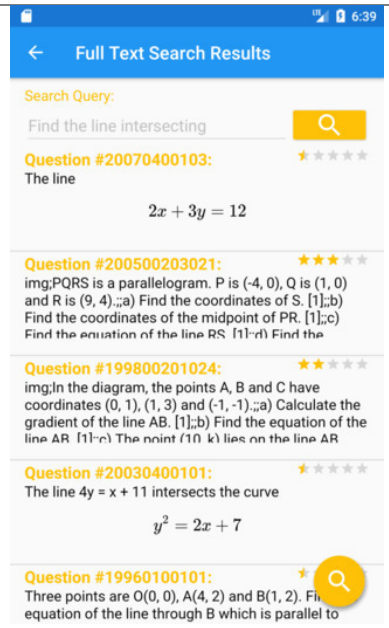
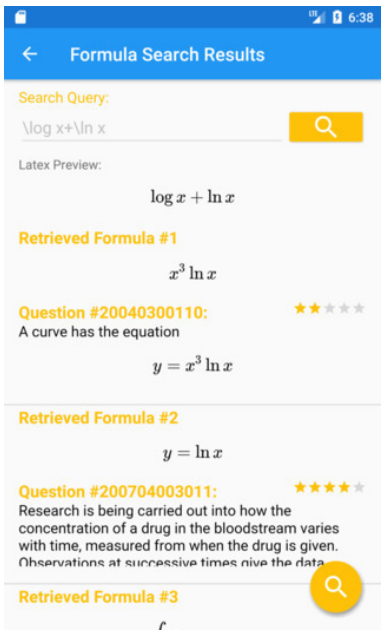


generator that is prompted when the `INSERT FORMULA` button is clicked.

(5) If the user chose to use the LaTeX symbol generator, a corresponding LaTeX syntax template for that particular symbol will get inserted automatically into the text editor. Users can edit this LaTeX syntax as necessary using the editable text editor.

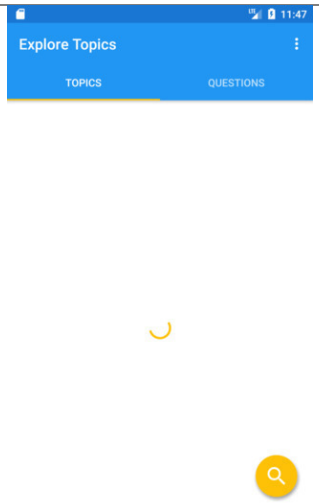
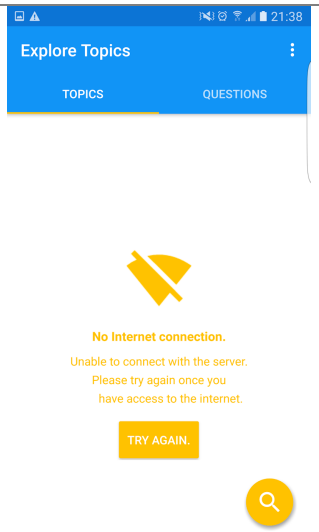
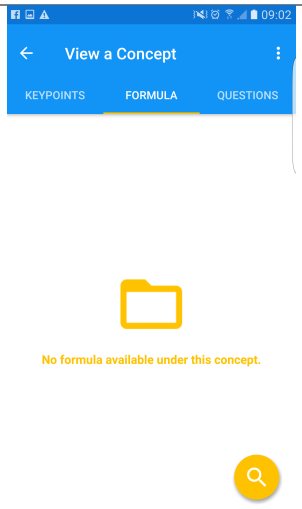
6.3.3.3 *Search Results*

Table 25. SearchResult Activity

User Interfaces	Activity
 	*QuestionSearchResultFragment* (1) The mathematical question search results are displayed as a simple ListView. (2) In this view, the latest query that the user had searched was also provided at the top of the fragment. When a user clicked on the search icon beside the previous query, the search dialog that corresponds to the latest query will be prompted. (3) For formula search results, the layouts are treated differently. Here, the related formula results will be displayed together with the question instead of just question results. A LaTeX preview of the latest formula query will also be displayed at the top of the fragment.

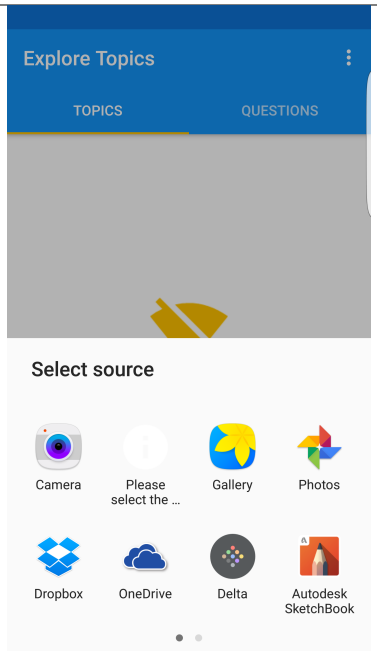
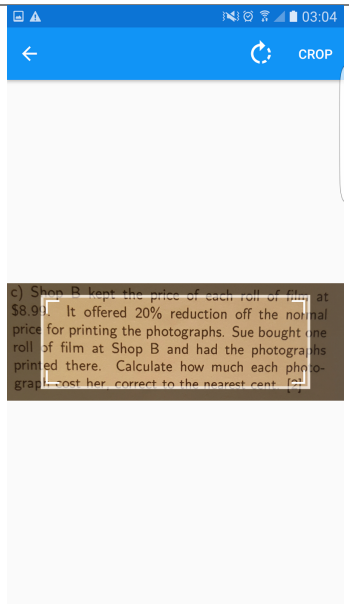
6.3.4 Progress Activities

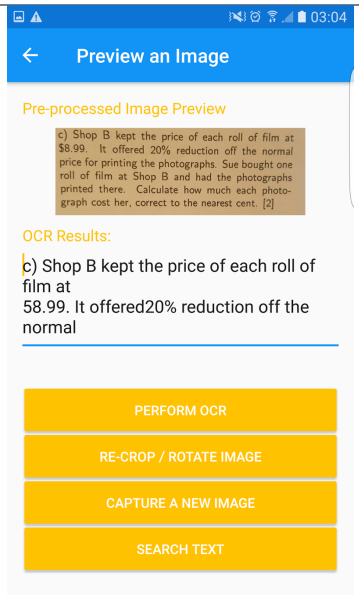
Table 26. Progress Activities

UI	Activity
	*Loading Activity* This activity indicates the loading state of the app.
	*Error No Connection* This activity will be shown when the app is not able to establish a connection with the server.
	*No Data* This activity shows the status of the UI when there is no content available from the server's response.

6.3.5 Document OCR Related UI

Table 27. Document OCR Related UI

User Interfaces	Activity
	*Image Picker* MathQA app allows users to select an image from the file system or camera image capture. To enable this, both storage and camera accesses are required at run time because the file Uniform Resource Identifier (URI) that points to the selected image is required for OCR.
	*Image Cropping Utilities* This utility allows the user to crop, zoom in/out, or rotate the selected image from the image picker.



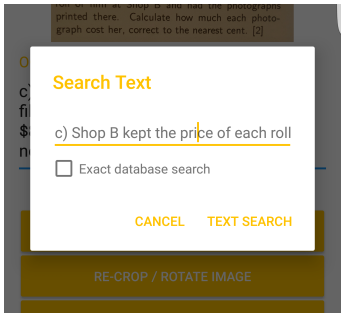


Image Preview Activity

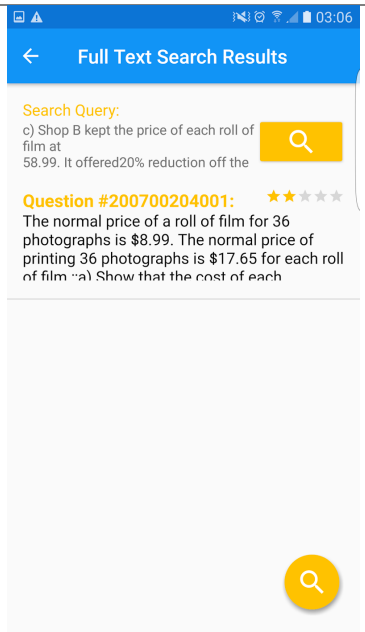
This activity displays the pre-processed image and the OCR results from OCR pipeline. This OCR result is returned in a form of an editable text field to allow users to modify the recognised content before performing a search. Further action buttons were also provided to the user in order to:

(1) Perform a text-based search using the recognised text.

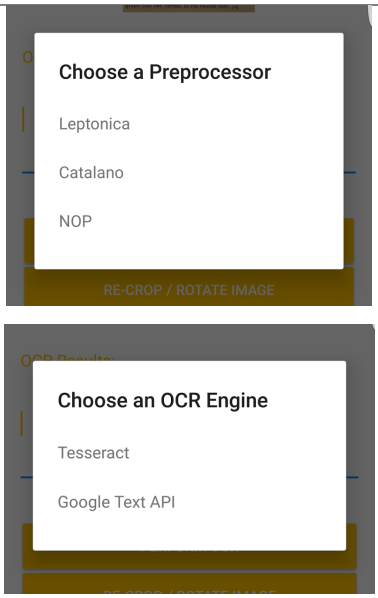
(2) Perform OCR on the cropped image using different OCR options.

(3) Re-crop or rotate the original image.

(4) Select / capture a new image source.



The image on the left is an example of the full-text search results returned by using the OCR result as the input to full-text search.



OCR Options Dialog

(1) These dialogs are prompted when the user clicked the `PERFORM OCR` button from ImagePreview activity.

(2) These dialogs ask users for their preferred image processor and OCR engine. Selected OCR options will then be passed to the OCR pipeline for image pre-processing and character recognition.

6.4 Discussions

During the Android development, author encountered many kinds of problems. However, through various experimentations and explorations, the author found that Android's vast open-source community had provided massive libraries and tools that can solve almost all the related problems in Android. From these findings, the author had succeeded in tackling all the Android-related problems for building MathQA and developed an ability to find, select, and incorporates various open-source tools and libraries to build a good app that can enrich the mathematical learning experience of the students.

At the end of Android development, the author has achieved the objective of designing and developing android app for learning mathematics. The features in the final MathQA app include:

(1) Accessing mathematical contents and performing mathematical document search through REST APIs.

(2) Viewing Mathematical contents: LaTeX rendering and intuitive and logical display between object models realised through ViewPagers, Fragments and ExpandableListViews.

(3) Supports for different modes of mathematical document search through two simple clicks from FloatingActionButtons and SearchDialogs.

(4) Camera capture and file storage access to select document image, performing OCR on the selected image, and use the OCR results to find mathematical documents.

Other than the app features, the following achievements were also made during the Android development.

(1) Initial system design plans, architecture and prototypes were meticulously designed to provide a maintainable app that adheres to good design principles and sustainable for future improvements. Figure 26 demonstrates how reusable view components were designed using object-oriented design principles. It demonstrates how similar view components are implemented by extending a base class. This base class abstracts the common properties and behaviours of the reusable components; therefore, the reusable component can derive from these base classes. This design adheres to the DRY and Liskov Substitution principle, where repetitive code is reduced due to object inheritance.

(2) Incorporated best design practises for building good user experience applications as specified in Material Design guideline. Various tools and libraries that support Material Design were used throughout the UI development.

7 DOCUMENT IMAGE ANALYSIS AND RECOGNITION

This chapter discusses the implementation details of Optical Character Recognition (OCR) service for Android. Experiments conducted to evaluate various OCR technologies were also discussed and analysed in this chapter.

7.1 Tools and Libraries

For the OCR implementation, three open-source libraries were experimented, namely: tess-two library [39] which include a Tesseract and Leptonica [18] tools for image processing in Android; Catalano [19], an image analysis and processing framework; and Google Text API [20].

7.2 Implementation

7.2.1 Obtaining the Image Source

Prior to performing document image recognition, camera and storage access must be allowed to obtain the image source. Image picker is provided to allow users to select an image source from camera or file storage. An image cropping and rotation feature were also incorporated so that once an image source is selected, users will be directed to a crop activity, where they can remove the image region from being recognised. All the image picking and cropping processes will generate an image uniform resource identifier (URI) which is utilised to locate the image source.

7.2.2 OCR Pipeline

Figure 27 depicts the OCR pre-processing pipeline that is designed and developed for MathQA. This pipeline is developed with flexibility in mind so that the Pipeline can accommodate different technologies of Image Pre-processors or OCR engines. Therefore, the addition or removal of a certain engine can be done without breaking other OCR-related components.

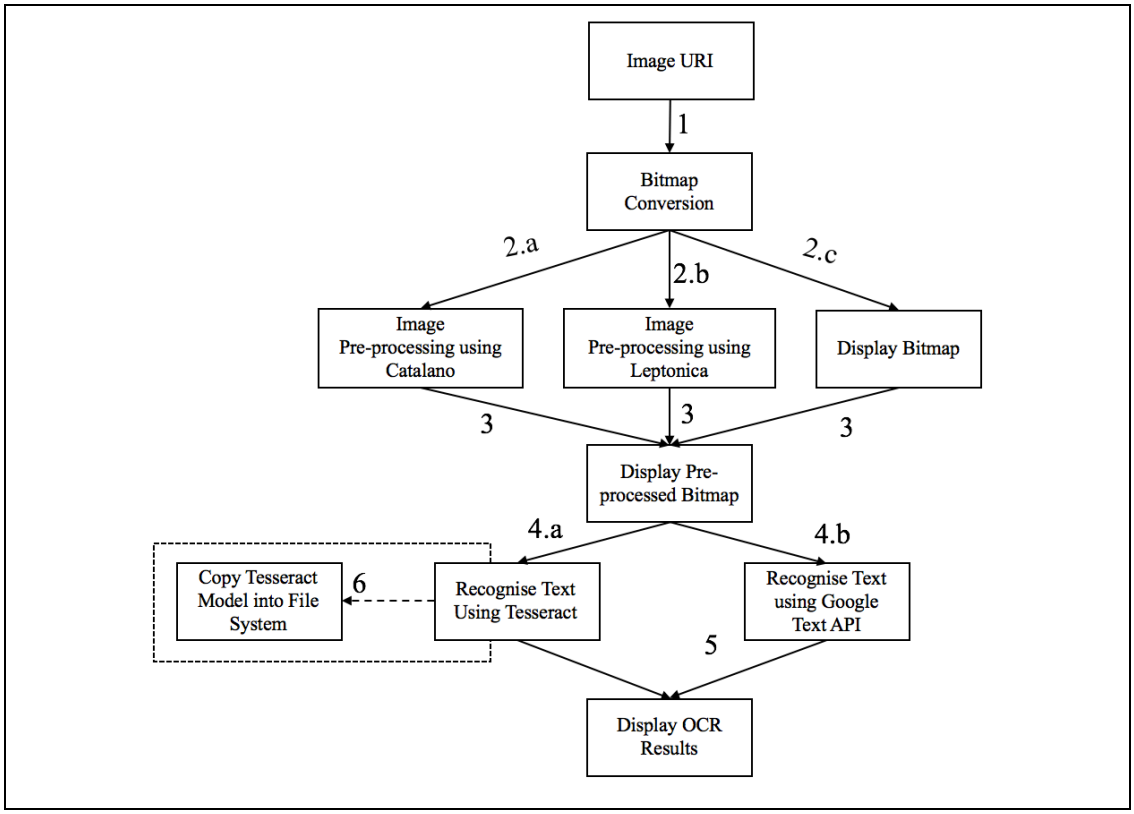


Figure 27. OCR Pipeline for MathQA

In the proposed pipeline, there were many paths to get OCR results from a source image through the different combinations of pre-processors and OCR engines. Notice that in step 6 from Figure 27, a training model for Tesseract is copied into mobile phone storage. This step is only required once when the Tesseract engine is used for the first time.

The branching at step 2 and 4 allows a flexible and extensible configuration for OCR recognition process. New OCR engines or image pre-processing tools can be added, modified or removed easily from the pipeline without affecting other existing components. This is achieved through object-oriented concepts: Abstraction and Polymorphism, where author attempted to abstract the common features of OCR processes into abstract OCR component base classes as displayed in Figure 28. New engines or pre-processors can be easily added by extending from these base classes.

This pipeline consists of four subtasks, which will be explained more in the following subsections.

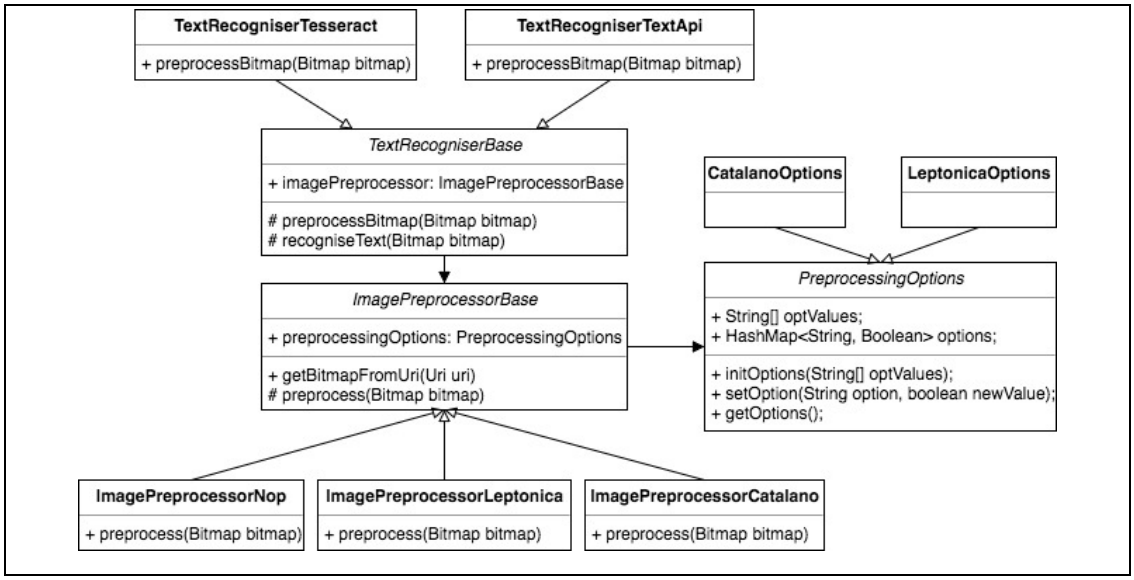


Figure 28. Class Diagram for OCR Related Components

7.2.2.1 Image Source Conversion to Bitmap

Bitmap corresponds to an image file format for storing digital images. This conversion is necessary because both Tesseract and Google Text API accept bitmap inputs. Prior to image conversion, the image source from the URI was resized to gain a better recognition performance. The reason for this stems from the fact that a larger image has too many features which could lower the OCR accuracy. Moreover, large data footprint can slow down the OCR pipeline processes, hence image downsizing step is necessary.

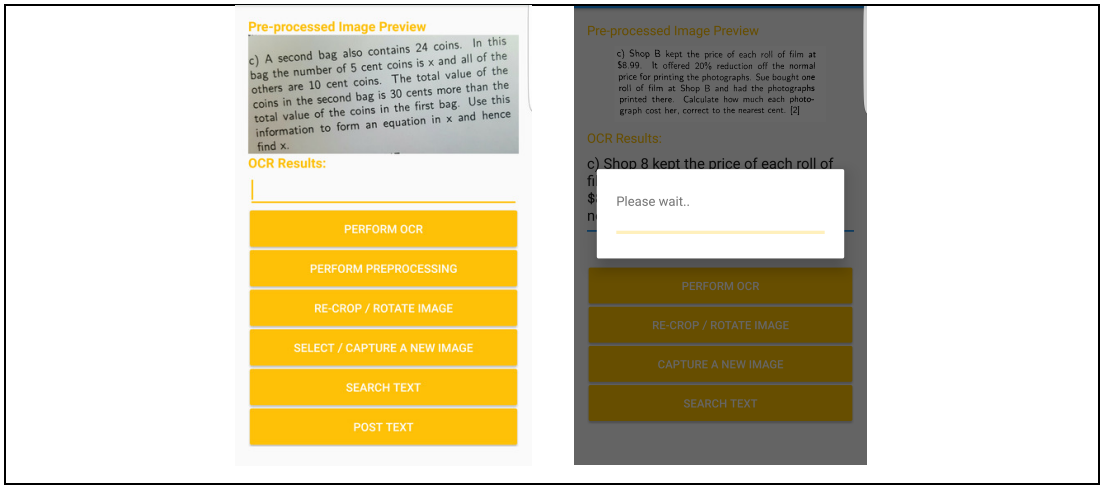


Figure 29. Example of Image Grayscaling (left) and Progress Loading during Pre-process and Recognition (right)

To resize the image, down sampling technique is utilised to compute the optimal sample ratio for downscaling. Note that this sample ratio is transformed into power of

two to gain a better sampling optimisation. See APPENDIX IV BITMAP PRE-PROCESSING for the detailed implementation.

7.2.2.2 Bitmap Image Pre-processing

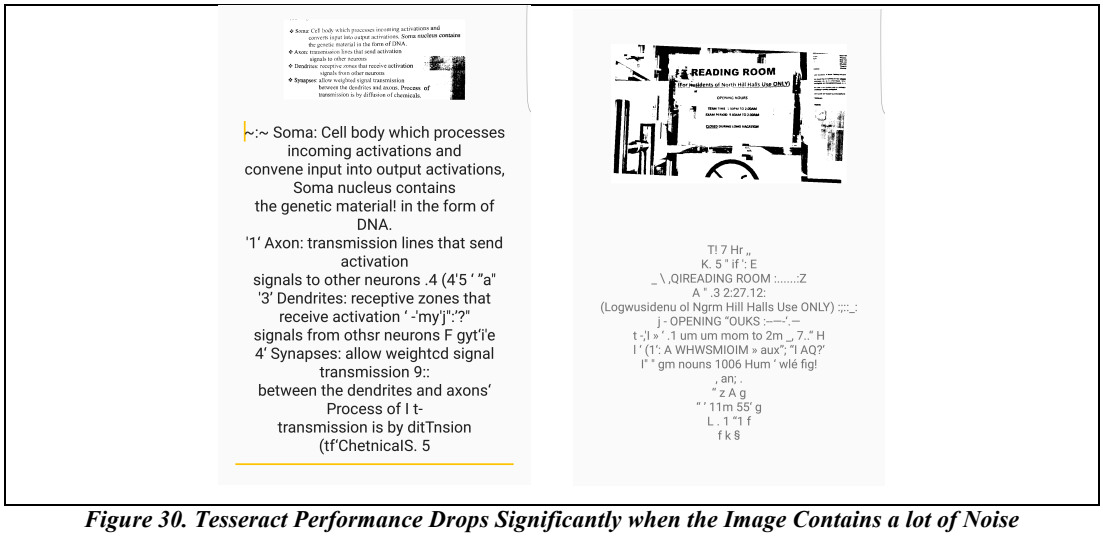


Figure 30. Tesseract Performance Drops Significantly when the Image Contains a lot of Noise

This step is especially required when Tesseract is used as the OCR engine. Generally, character recognition performs quite well when the image is taken with good lighting and the contrast between text and background is clear. However, OCR engine performance may drop significantly when there are a plenty of noises available in the image as demonstrated in Figure 30. Therefore, image pre-processing methods were added into the OCR pipeline to pre-process bitmaps for improving Tesseract performance during recognition. The rescaled bitmap obtained from step 7.2.2.1 was then passed into an image pre-processor.

In order to select the correct pre-processing actions, some experiments were conducted to observe the capability of the existing pre-processors in cleaning the document image. This is made possible by altering the Android UI a bit to cater dynamic selection of pre-processing actions as shown in Figure 31.

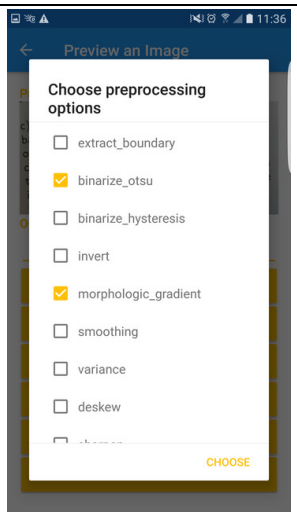


Figure 31. Selecting Pre-processing Actions Dynamically through Android UI

Based on the experimental results, these four pre-processing action combinations were identified as produce a fine quality pre-processed bitmap. These actions include: (1) contrast and (2) background normalisation, (3) binarisation using sauvola's technique, and (4) image de-skewing.

7.2.2.3 OCR on Pre-processed Bitmap

After the image pre-processing is completed, the pre-processed Bitmap is then sent to the OCR engine for character recognition. After this step is completed, the OCR result will get displayed in an editable text box where the user can edit before they perform a mathematical document retrieval (see section 6.3.5 for the related views).

7.2.3 Best Practices

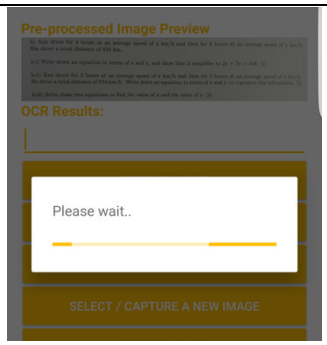


Figure 32. UI Blocking during Image Pre-processing and OCR

Image pre-processing and OCR processes are heavy processes; therefore, these two operations must be run in the background thread. A blocking mechanism was provided in the UI to block user actions until the image pre-processing and OCR

processes are finished. Through this blocking mechanism, user interference during the recognition process can be prevented.

7.3 Testing and Analysis

7.3.1 OCR Engine Configurations

To select the best OCR tool for MathQA, an experiment was carried out to evaluate the different OCR engines and image pre-processors performance. Four different OCR configurations experimented:

(1) Tesseract without image pre-processing (NOP).
(2) Tesseract with image pre-processing using Leptonica.
(3) Tesseract with image pre-processing using Catalano Framework.
(4) Google Text API without image pre-processing.

7.3.2 Test Image Preparation

Several mathematical question images were picked from the database and printed to observe the OCR performance in recognising mathematical document image. The images that had been chosen for testing in this report have varying characteristics:

(1) Image with Sans-serif font content. Sans-serif is a category of typeface that do not have serifs (i.e. tiny lines at the ends of characters). This type-face is used as the default type-face during the testing because it has a higher accuracy than serif images.
    i. A skewed image that contains only letters and numbers.
    ii. Normal image that contains only letters and numbers.
    iii. Normal image that contains letters, numbers and some symbols such as currency and percentage.
    iv. Normal image with LaTeX.
(2) Image with Serif font content. Serif is a typeface that has an extended feature at the end of the stroke.
    v. Normal image with mostly text data.
    vi. Normal Image with LaTeX.

7.3.3 Tools

To perform OCR testing, three tools were used. MathQA app was utilised to get the OCR results from the document image and post the OCR results to the server,

Django was used to receive the text content from the user's POST request, and a simple Python script was run to compute the precision and recall.

7.3.4 Evaluation Methods

Based on the four OCR configuration proposed in 7.3.1, author then performed OCR on the test images with the varying characteristics specified in step 7.3.2. The text results obtained from the OCR are then evaluated to calculate the precision, recall and processing time of each OCR configuration. Time here is defined as the time taken from user's request for OCR until the OCR text result is returned to the user.

The formulas used to evaluate the OCR performance are as follows:

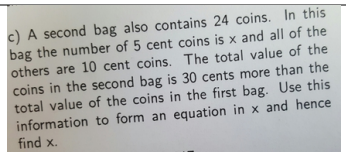
Table 28. Precision and Recall Formulas for OCR Evaluation

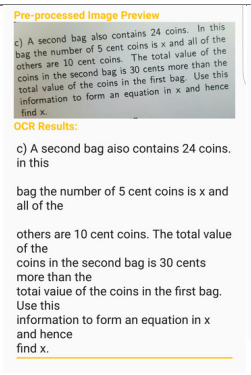
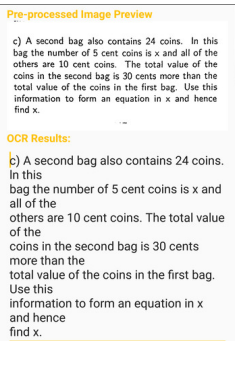
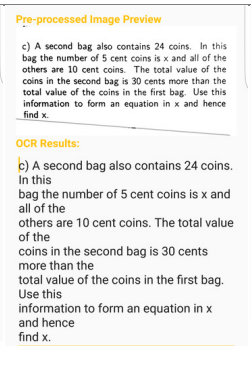
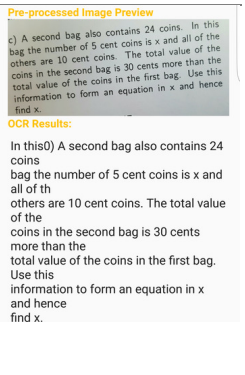
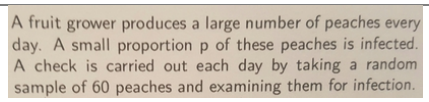
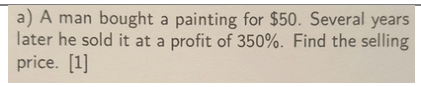
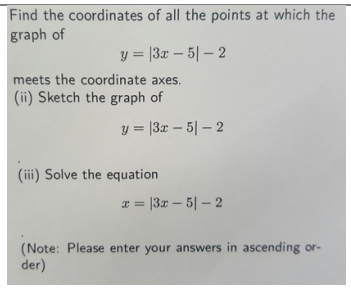
Formula	Definitions
$\text{Precision} = \frac{tp}{tp + fp}$ $\text{Recall} = \frac{tp}{tp + fn}$	tp: true positives, the number of correct characters recognised by the ocr. fp: false positives, the number of characters recognised by the ocr but do not exist in the source image. fn: false negatives, the number of characters in the source image that is not recognised by the ocr.

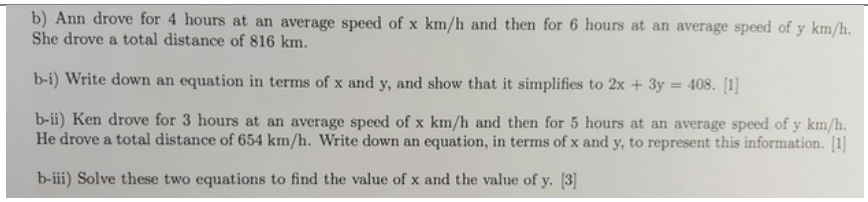
7.3.5 Results and Analysis

The following tables, Table 29 and Table 30, display the OCR experimental results.

Table 29. OCR Experimental Results

(1) Skewed sans-serif image			
			
Tesseract + NOP	**Tesseract + Leptonica**	**Tesseract + Catalano**	**Google Text API + NOP**

Results				
Precision	0.99	0.996	1.0	0.995
Recall	0.974	0.996	1.0	0.99
Speed (s)	1.62	1.56	8.12	1.3
(2) Sans-serif image that contains only letters and numbers				
				
Precision	1.0	1.0	1.0	1.0
Recall	1.0	1.0	1.0	0.99
Time (s)	0.96	1.07	4.2	0.967
(3) Sans-serif image that contains letters, numbers and some symbols				
				
Precision	1.0	1.0	1.0	1.0
Recall	1.0	1.0	0.967	0.967
Time (s)	0.65	1.151	3.01	0.94
(4) Sans-serif image that contains LaTeX string				
				
Precision	0.952	0.967	0.962	0.969
Recall	0.836	0.831	0.83	0.752
Time (s)	1.352	1.953	8.77	1.47
(5) Serif image that contains mostly text data				


b) Ann drove for 4 hours at an average speed of x km/h and then for 6 hours at an average speed of y km/h. She drove a total distance of 816 km.

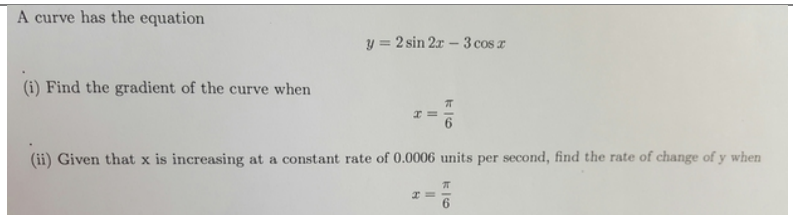
b-i) Write down an equation in terms of x and y, and show that it simplifies to $2x + 3y = 408$. [1]

b-ii) Ken drove for 3 hours at an average speed of x km/h and then for 5 hours at an average speed of y km/h. He drove a total distance of 654 km/h. Write down an equation, in terms of x and y, to represent this information. [1]

b-iii) Solve these two equations to find the value of x and the value of y. [3]

Precision	0.874	0.917	0.83	0.983
Recall	0.794	0.88	0.75	0.981
Time (s)	11.55	10.12	9.61	0.907

(6) Serif image with LaTeX


A curve has the equation

$$y = 2\sin 2x - 3\cos x$$

(i) Find the gradient of the curve when

$$x = \frac{\pi}{6}$$

(ii) Given that x is increasing at a constant rate of 0.0006 units per second, find the rate of change of y when

$$x = \frac{\pi}{6}$$

Precision	0.83	0.87	0.79	0.967
Recall	0.70	0.78	0.703	0.832
Time (s)	4.5	3.5	5.64	0.73

Table 30. Overall Average OCR Evaluation Results

Metrics	**Tesseract + NOP**	**Tesseract + Leptonica**	**Tesseract + Catalano**	**Google Text API**
Average Precision	0.941	0.96	0.93	0.986
Average Recall	0.884	0.9145	0.88	0.922
Average Time (s)	2.857	2.76	5.93	1.074

Based on the test results in Table 29, it can be observed that in general, Tesseract without image pre-processing, Tesseract with image pre-processing with Leptonica, and Google Text API all have decent performance in recognising text from a document image. They have similar fast average processing time of less than 3 seconds. The Catalano framework however, found to have poor performance due to its slow pre-processing time.

For the skewed image in image test (1), better results were observed when the image was pre-processed. Although Google Text API was able to recognise the content, the order where the recognised text is placed in the result is incorrect. This situation occurs because Google Text API uses TextBlocks (similar to bounding box) to detect texts from document image. In the results, these blocks are sorted according

to the top left corner position of the TextBlocks. An example of this can be seen from the phrase `in this` image in test (1). This phrase is contained in the top most TextBlock from the image, therefore it gets displayed first instead of `c) A second bag`.

Fonts used in the document image affects the performance of an OCR engine. As observed from the test results, the OCR performance for Tesseract, in particular, was dropped when serif fonts are used in serif image test (5). In this particular test, Google Text API outperforms all other OCR configurations in terms of precision and speed.

7.4 Discussions

Based on the implementation and test results, the following points can be made:

(1) Image pre-processing is an important process during OCR. However, based on author's experiments, author found that when a document image is taken proper lighting and contrast, both Tesseract and Google Text API without image pre-processing are robust enough recognise the full text content with a fast processing time.

(2) Catalano can be removed from the OCR pipeline because it has poor performance in terms of pre-processing image and processing time.

(3) Tesseract without pre-processor can outperform all other OCR configurations when only alphanumerical symbols appear in a text that used sans-serif fonts, and the image has good lighting conditions. However, it performed worse when the image contains many noises, non-alphanumerical symbols or when serif fonts is used in the document image.

(4) The drawback of using tesseract is that it requires the training data model to be copied onto the phone. Hence it increases the app data footprint. Google Text API on the other hand does not require any additional installation or copying models into the phone.

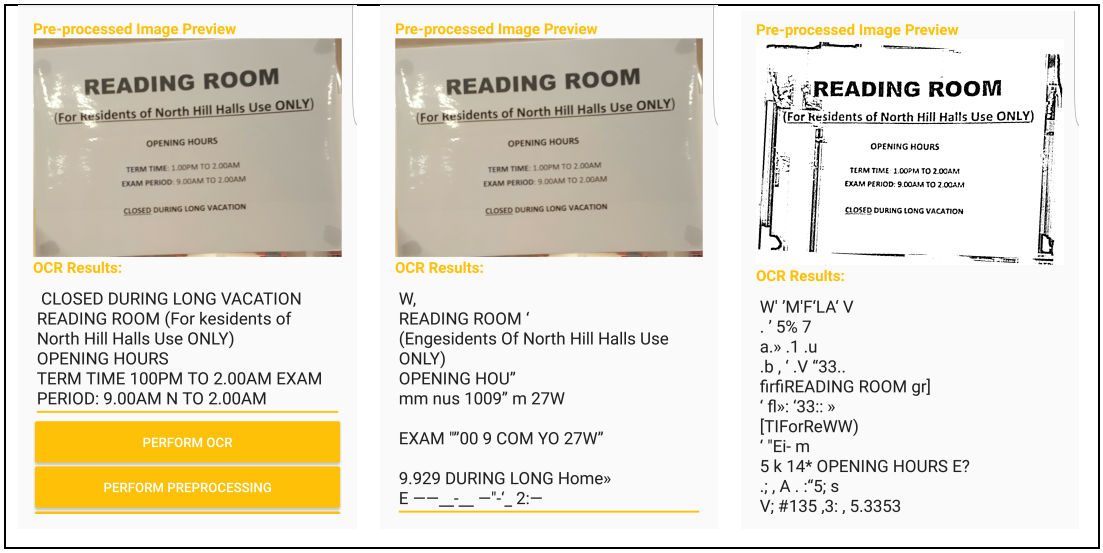


Figure 33. Google Text API (left), Tesseract (middle and right) Performance in Recognising a Poor-quality Image

Based on the overall test results in Table 29 and Table 30, Google Text API was identified to have the best performance among all OCR tools in recognising content with a high precision and recall from all images of all conditions. Therefore, Google Text API is used as the default OCR engine in MathQA app. The drawback of this API however, is that it might not return the results in the correct order when the image is not properly aligned. This can be easily fixed by de-skewing the image prior to recognition by the Google Text API engine.

8 CONCLUSION AND FUTURE WORKS

8.1 Conclusion

MathQA system is a web-based learning platform that helps students solving mathematical problems. As part of this system, a mobile application was developed to allow easy access to the MathQA services from anywhere at any time.

There are two major tasks identified in order to build web-based mathematical document retrieval for mobile application. First, is to build the database model and services for retrieving mathematical documents and second is to develop the Android application that utilises these services.

In the first task, database and mathematical document retrieval services were developed for the MathQA server using Django framework. These services were then exposed as a set of REST APIs which is consumed by the Android app. Three modes of mathematical document retrieval services were provided in the app. These techniques include database query filter, full-text, and formula-based retrievals. Formula-based retrieval is the most interesting among the three retrievals because mathematical formulas are complex. An inverted-index technique for formula-based retrieval proposed in [1] and [2] were investigated, implemented, and evaluated. Based on the evaluation results, the proposed algorithm achieved a promising performance of having 97.5% MAP@10 and Average P@10 of 91.4%.

Next, in the second task, Android app for MathQA was developed. The Android app is able to display mathematical contents from MathQA through clean and intuitive user interfaces that adhere to good design practices specified in Material Design Guideline. To enhance the user experience when searching for a mathematical content, an Optical Character Recognition (OCR) feature was also incorporated into the app. Through this OCR service, MathQA users are able to perform a mathematical document search from a simple image capture.

Several experiments were first conducted to assess the performance of different OCR tools. From the experimental results, it is observed that both Tesseract and Google Text API are fast and very good at recognising textual contents with high precision and recall. Google Text API in particular, is found to be the best performing

among all engines as it is able to recognise document images without the need of an image pre-processor.

Aside from building good user interface intended for good user experience, software engineering principles were also taken into account during the web-based and Android app development to build software that is maintainable, extensible and sustainable for future developers. From the process and output of the project, it can be concluded that the author has achieved the project objective of developing a web-based mathematical document retrieval system for a mobile app that can facilitate learning mathematics.

8.2 Future Recommendations

Author has identified several recommendations to enhance MathQA performance and functionalities. Future developers that will continue this project can adopt the following recommendations to improve the existing project.

8.2.1 Mathematical Document Retrieval

(1) The mathematical document retrieval used in this project separates the question content into two parts: text and formula. Therefore, some information might be lost due to this content separation. Mathematical document retrieval techniques that can combine these two results can be investigated to further enhance the mathematical document retrieval performance.

(2) The current implementation of formula-based retrieval is heavily dependent on the ability of the MathML parser. Currently, a third-party library latex2mathml [24] is used in the system. However, this library has some limitations as discussed in section 5.3.2.5. Different methods for obtaining mathematical structure from LaTeX can be explored to further improve the formula-based retrieval.

(3) Author's initiative in building admin GUI had freed the developers from having to manually inspect the textual representation of mathematical contents. This GUI can be used for experimenting with the new mathematical document retrieval techniques or extended to support advanced techniques for evaluating mathematical document retrieval performance.

8.2.2 Document Image Recognition

The OCR implemented in this project was only able to detect alphanumeric characters and unable to detect LaTeX syntax from a document image. Employing

advanced techniques for automatic LaTeX recognition such as deep learning methods will be useful. Since the author had developed a flexible OCR pipeline, future OCR engine improvements can be added seamlessly into the current MathQA app.

8.2.3 Extending MathQA Features

The current MathQA mobile app development initiated in this project was focused on providing a read-only access to the mathematical contents and performing robust mathematical document retrieval services. A dynamic and interactive learning platform where users can interact with their teachers or other students, a full forum where people can discuss and ask and answers questions can be developed on top of the current app to enhance student's learning experience.

8.2.4 Conduct a Usability Study

Usability testing can identify usability problems and determine user's satisfaction regarding the MathQA app. This study can be done as follows. First, a group of students can be selected (say, 15-20) to participate in a usability study. Next, these students are asked to complete some tasks that address different aspects of the app. Then, an evaluation is done to see where the students encounter problems and experience confusion and recorded as a guideline for improving MathQA.

BIBLIOGRAPHY

[1] M. Kai, “Data Mining for Mathematical Question Answering Community,” Singapore, 2011.

[2] S. H. Samarasinghe, “Semantic-based Retrieval for Mathematical Knowledge,” Singapore, 2009.

[3] LaTeX3 Team, “The LaTeX Project,” [Online]. Available: http://www.latex-project.org/about/. [Accessed 2017].

[4] W3C, “Mathematical Markup Language (MathML) Version 3.0 2nd Edition,” [Online]. Available: https://www.w3.org/TR/MathML3/.

[5] C. D. Manning, R. Prabhakar and S. Schütze, Introduction to Information Retrieval, New York: Cambridge University Press, 2008.

[6] DjangoGirls, “Django Girls Tutorial,” [Online]. Available: https://tutorial.djangogirls.org/en/django/. [Accessed 2017].

[7] Google Inc, “Angular Material,” [Online]. Available: https://material.angularjs.org/latest/. [Accessed 2017].

[8] W. S. Jawadekar, Knowledge Management: Text & Cases, New Delhi: Tata McGraw-Hill Education Private Ltd, 2012, p. 278.

[9] Black Duck Software Inc, “OpenHub Android,” 2017. [Online]. Available: https://www.openhub.net/p/android.

[10] Google Inc, “Android for Developers,” [Online]. Available: https://developer.android.com.

[11] N. Smyth, Android Studio 2.3 Development Essentials, Payload Media, 2017.

[12] S. Aghav and S. Paygude, “An Approach for Document Image Based Printed Character Recognition,” in *Proceedings of ICAdC*, 2013.

[13] R. Fisher, S. Perkins, A. Walker and E. Wolfart, Hypermedia Image Processing Reference, 2000.

[14] M. Elmore and M. Martonosi, “A Morphological Image Preprocessing Suite for OCR on Natural Scene Images,” 2008.

[15] L. O'Gorman and R. Kasturi, "Document Image Analysis," 2009.

[16] W. Bieniecki, S. Grabowski and W. Rozenberg, "Image Preprocessing for Improving OCR Accuracy," in *Perspective Technologies and Methods in MEMS Design*, 2007.

[17] R. Smith, "An Overview of the Tesseract OCR Engine".

[18] D. Bloomberg, "Leptonica," 2017. [Online]. Available: http://www.leptonica.com/.

[19] D. Catalano, "Catalano Framework," [Online]. Available: https://github.com/DiegoCatalano/Catalano-Framework. [Accessed 2017].

[20] Google Inc, "Text Recognition API Overview," [Online]. Available: https://developers.google.com/vision/text-overview. [Accessed 2017].

[21] R. C. Martin, Design Principles and Design Patterns, 2000.

[22] A. Hunt and D. Thomas, The Pragmatic Programmer, 1999.

[23] T. Christie, "Django REST Framework," 2017. [Online]. Available: http://www.django-rest-framework.org/.

[24] R. Martinez, "Pure Python library for LaTeX to MathML conversion.," [Online]. Available: https://github.com/Code-ReaQtor/latex2mathml. [Accessed 2017].

[25] D. Lindsley, "Welcome to Haystack! — Haystack 2.5.0 documentation," [Online]. Available: http://django-haystack.readthedocs.io/en/v2.6.0/index.html.

[26] M. Chaput, "Whoosh 2.7.4 documentation," 2017. [Online]. Available: https://whoosh.readthedocs.io/en/latest/.

[27] NLTK Project, "Natural Language Toolkit," 2015. [Online]. Available: https://www.nltk.org/.

[28] Khan Academy, "Fast math typesetting for the web," [Online]. Available: https://github.com/Khan/KaTeX. [Accessed 2017].

[29] Code-ReaQtor, "Pure Python library for LaTeX to MathML conversion," [Online]. Available: https://github.com/Code-ReaQtor/latex2mathml.

[30] AoDevBlue, "An Android library for material design values," 2016. [Online]. Available: https://aodevblue.github.io/MaterialValues/.

[31] D. Steduto, “Fast and versatile Adapter for your RecyclerView,” 2017. [Online]. Available: https://github.com/davideas/FlexibleAdapter/.

[32] D. Tarianyk, “FloatingActionButton,” [Online]. Available: https://github.com/Clans/FloatingActionButton. [Accessed 2017].

[33] A. Follestad, “A beautiful, fluid, and customizable dialogs API.,” [Online]. Available: https://github.com/afollestad/material-dialogs. [Accessed 2017].

[34] V. Gashi, “Easily add loading, empty and error states in your app.,” [Online]. Available: https://github.com/vlonjatg/progress-activity. [Accessed 2017].

[35] B. Lee, “MathView, A library for displaying math formula in Android apps.,” [Online]. Available: https://github.com/kexanie/MathView. [Accessed 2017].

[36] Square Inc., “Type-safe HTTP client for Android and Java by Square, Inc.,” [Online]. Available: https://github.com/square/retrofit. [Accessed 2017].

[37] RxJava Contributors, “RxJava2,” [Online]. Available: https://github.com/ReactiveX/RxJava/tree/2.x. [Accessed 2017].

[38] A. Teplitzki, “Image Cropping Library for Android, optimized for Camera / Gallery.,” [Online]. Available: https://github.com/ArthurHub/Android-Image-Cropper. [Accessed 2017].

[39] R. Theis, “Tesseract Tools for Android,” [Online]. Available: https://github.com/rmtheis/tess-two. [Accessed 2017].

[40] J. Ericksen, “Android Parcelables made easy through code generation.,” [Online]. Available: https://github.com/johncarl81/parceler. [Accessed 2017].

[41] K.-U. Janssen, “Fast Android Development. Easy maintainance.,” [Online]. Available: https://github.com/androidannotations/androidannotations/wiki/. [Accessed 2017].

[42] Google, Inc., “Fragments | Android Developers,” [Online]. Available: https://developer.android.com/guide/components/fragments.html. [Accessed 15 03 2017].

APPENDIX I AVAILABLE REST API

AI.1 Available REST API from the Server

```
from apiv2 import views

router = routers.SimpleRouter()

router.register(r'subjects', views.SubjectViewSet)
router.register(r'topics', views.TopicViewSet)
router.register(r'concepts', views.ConceptViewSet)
router.register(r'subconcepts', views.SubconceptViewSet)
router.register(r'papersets', views.PapersetViewSet)
router.register(r'papers', views.PaperViewSet)
router.register(r'questions', views.QuestionViewSet)
router.register(r'solutions', views.SolutionViewSet)
router.register(r'formulas', views.FormulaViewSet)
router.register(r'formula_categories', views.FormulaCategoryViewSet)
router.register(r'formula_indexes', views.FormulaIndexViewSet)
router.register(r'keypoints', views.KeyPointViewSet)
router.register(r'keywords', views.KeywordViewSet)

urlpatterns = [
    url(r'^', include(router.urls)),
    url(r'^search/$', views.search),

    url(r'^reindex_all_formula/$', views.reindex_all_formula),
    url(r'^create_update_formula/$', views.create_update_formula),
    url(r'^delete_formula/$', views.delete_formula),
    url(r'^check_mathml/$', views.check_mathml_str),
    url(r'^check_formula_token/$', views.check_formula_token),

    url(r'^update_question/$', views.update_question),
    url(r'^update_solution/$', views.update_solution),
    url(r'^update_keypoint/$', views.update_keypoint),
]

urlpatterns = format_suffix_patterns(urlpatterns)

urlpatterns += [
    url(r'^api-auth/',
        include('rest_framework.urls', namespace='rest_framework')),
    url(r'^api-token-auth/', rest_views.obtain_auth_token),
    url(r'^auth/', include('djoser.urls.authtoken')),
    url(r'^login/', auth_views.login),
    url(r'^logout/', auth_views.logout),
]
```

Explanations:

(1) A router registers view handlers which supports `READ ONLY ACCESS` to the database. Example of how a router handle listing and retrieving data object:

/subjects: retrieve all subjects

/subjects/<Pk>: get subject by ID.

(2) Search is an API that is available via HTTP GET method. The presence of query and type in the URL ?type=[type] and ?query=[query] is required to provide the search type and search query to the server. There are three search types available:

i. d: database search
ii. f: formula search
iii. t: full-text search

(3) APIs that manipulates the database, such as reindexing formulas, create, update, and delete operations on formula, questions, solutions, and keypoints are available via HTTP POST, PATCH. These requests are considered dangerous therefor, an admin-level access is required.

AI.2 Available Query Filters from the Server

Django Model	**Filter By**
Topic	Subject /topics/?subject=<subject_id>: retrieve topics by subject
Concept	Topic /concepts/?topic=<topic_id>: retrieve concepts by topic
PaperSet	Subject /paperset/?subject=<subject_id>: retrieve paperset by subject
Paper	Paperset /papers/?paperset=<paperset_id>: retrieve papers by paperset
Question	Concept /questions/?concept=<concept_id>: retrieve questions by concept Subconcept /questions/?subconcept=<subconcept_id>: retrieve questions by subconcept Paper /questions/?paper=<paper_id>: retrieve questions by paper id Keypoints /questions/?keypoints=<keypoint_id>: retrieve questions by keypoint Keywords /questions/?keywords=<keyword_id>: retrieve questions by keywords Formula_Categories

	/questions/?formula_categories=<formula_category_name>: retrieve questions by formula categories
Solution	Question /solutions/?question=<keyword_id>: retrieve solutions by question
Formula	Questions /formulas/?question=<question_id>: retrieve formulas by question Categories /formulas/?category=<formula_category_id>: retrieve formulas by formula category
Keypoint	Concept /keypoints/?concept=<concept_id>: retrieve questions by concept Type /keypoints/?type=C: retrieve theoretical background of a concept /keypoints/?type=F: retrieve key formulas of a concept
Keyword	Questions /keywords/?questions=<question_id>: retrieve keywords by question

AI.3 Available Android Client REST APIs

REST API	**Actions**
getSubjects()	Request all subjects from the server and store it as List<Subject>
getTopics()	Request all topics and store it as List<Topic>
getTopics(subjectId)	Request all topics under a subject and store it as List<Topic>
getConcepts(topicId)	Request all concepts under a topic and store it as List<Concept>
getSubconcepts(conceptId)	Request all subconcepts under a concept and store it as List<Subconcept>
getKeypointConcepts(conceptId)	Request all theoretical keypoints under a concept and store it as List<Keypoint>

getKeypointFormulas(conceptId)	Request all formula keypoints under a concept and store it as List<Keypoint>
getQuestions()	Request all questions and store it as List<Question>
getQuestions(conceptId)	Request all questions under a concept and store it as List<Question>
getQuestions(subconceptId)	Request all questions under a subconcept and store it as List<Question>
getSolutionsByQuestion(questionId)	Request for solution of a question and stores it as List<Solution>

APPENDIX II FORMULA TERMS

AII.1 Supported Formula Function Terms

1	arccos	11	gcd	20	log	29	tanh
2	cos	12	lg	21	sec	30	int
3	csc	13	ln	22	tan	31	sum
4	exp	14	sup	23	arg	32	sqrt
5	limsup	15	arctan	24	coth	33	frac
6	min	16	cot	25	dim	34	sum
7	sinh	17	det	26	inf	35	∑
8	arcsin	18	lim	27	liminf	36	∫
9	cosh	19	sin	28	max	37	√
10	deg						

AII.2 Supported Formula Operator Terms

1	pm	15	times	29	div
2	ast	16	dot	30	cap
3	cup	17	vee	31	wedge
4	+	18	-	32	times
5	plus	19	±	33	×
6	÷	20	−	34	⁄
7	∖	21	∗	35	∘
8	·	22	∝	36	∞
9	∟	23	∠	37	//
10	∧	24	∨	38	∩
11	∪	25	leq	39	geq
12	equiv	26	nequiv	40	models
13	sim	27	simeq	41	mid
14	parallel	28	subset	42	supset

43	approx	73	subseteq	103	supseteq
44	cong	74	join	104	neq
45	proptoin	75	=	105	<
46	>	76	≠	106	≡
47	≠	77	≡	107	≤
48	≥	78	≦	108	≧
49	≃	79	⊂	109	⊃
50	⊄	80	⊅	110	⊆
51	⊇	81	⊈	111	⊉
52	⊊	82	⊋	112	⋈
53	|	83	-:	113	~
54	∽	84	≂	114	≃
55	≠	85	≅	115	≇
56	≉	86	≈	116	≠
57	≋	87	≋	117	≌
58	!	88	,	118	colon
59	ldotp	89	cdotp	119	∴
60	⋱	90	:	120	::
61	leftarrow	91	Leftarrow	121	rightarrow
62	Rightarrow	92	leftrightarrow	122	Leftrightarrow
63	leftharpoonup	93	leftharpoondown	123	rightleftharpoons
64	rightharpoonup	94	rightharpoondown	124	mapsto
65	ldots	95	cdots	125	vdots
66	ddots	96	forall	126	infty
67	exists	97	nabla	127	neg
68	triangle	98	angle	128	bot
69	prime	99	emptyset	129	…
70	prod	100	coprod	130	bigcap
71	bigcup	101	bigodot	131	bigotimes
72	bigoplus	102	(	132	)

133	uparrow	144	Uparrow	155	[
134	]	145	downarrow	156	Downarrow
135	{	146	}	157	updownarrow
136	Updownarrow	147	lfloor	158	rfloor
137	lceil	148	rceil	159	langle
138	rangle	149	/	160	backslash
139	|	150	lgroup	161	rgroup
140	\	151	.	162	'
141	widetilde	152	widehat	163	overleftarrow
142	overrightarrow	153	overline	164	underline
143	overbrace	154	underbrace		

APPENDIX III FULL FORMULA SEARCH EVALUATION RESULTS

No.	Formula Categories	Test Queries	P{5}	P{10}	AP{5}	AP{10}
1	Absolute	|x - 1|	1.00	1.00	1.00	1.00
2	Absolute	|x^{2}-8x+7|	1.00	0.80	1.00	0.92
3	Absolute	|x - 1| + |x - 3|	1.00	1.00	1.00	1.00
4	Absolute	3y=|4x-8|	0.80	0.90	0.95	0.91
5	Absolute	|1 + x| < 3	1.00	0.80	1.00	0.92
6	Exponential	e^{x}	1.00	1.00	1.00	1.00
7	Exponential	3^{x}	1.00	1.00	1.00	1.00
8	Exponential	3^{y+1}	1.00	0.80	1.00	0.98
9	Exponential	e^{x} - e^{-x}	0.80	0.80	1.00	0.93
10	Exponential	xe^{2x}	1.00	1.00	1.00	1.00
11	Inequality	4 - x < 6	1.00	1.00	1.00	1.00
12	Inequality	-8 < 2x + 3 < 11	1.00	1.00	1.00	1.00
13	Inequality	20-3y < y+4	0.80	0.90	1.00	0.93
14	Inequality	x^{2}+7x-9<8x-3	1.00	1.00	1.00	1.00
15	Inequality	x+1 < 7 < x+3	0.80	0.40	1.00	1.00
16	Fraction	1 + \frac{1}{x}	1.00	0.90	1.00	0.98
17	Fraction	\frac{12}{6-x}	1.00	0.80	1.00	0.99
18	Fraction	\frac {3 - 2x}{x - 2}	0.80	0.70	0.95	0.86
19	Fraction	\frac{ s^{2}+2 }{ s+1 }	1.00	1.00	1.00	1.00
20	Fraction	\frac{12}{x+3}	1.00	1.00	1.00	1.00
21	Integral	\int_{\frac{\pi}{2}}^{\pi} \cos^{2} x \mathrm{d}x	1.00	0.90	1.00	1.00
22	Integral	\int_{0}^{1} xe^{2x} \mathrm{d}x	1.00	1.00	1.00	1.00
23	Integral	\int \frac{2}{1+3x} \mathrm{d}x	1.00	0.80	1.00	1.00
24	Integral	\int_{0}^{3} x\sqrt {x + 1} \mathrm{d}x	1.00	1.00	1.00	1.00
25	Integral	\int x^{2} e^{3x} \mathrm{d}x	1.00	1.00	1.00	1.00
26	Line	x + y = 1	0.80	0.80	0.68	0.77
27	Line	y = x + 1	0.80	0.80	1.00	0.91
28	Line	x^{3}	1.00	0.90	1.00	0.99

29	Line	y = 2x^{2} -6x +c	1.00	0.80	1.00	0.95
30	Line	y=\frac{1}{2}x+2	1.00	0.80	1.00	0.99
31	Logarithms	\lg a	1.00	1.00	1.00	1.00
32	Logarithms	\log_{3}(x-1)	1.00	1.00	1.00	1.00
33	Logarithms	\lg x + 2\lg y	1.00	1.00	1.00	1.00
34	Logarithms	2 + \lg (2x + 1)	1.00	0.80	1.00	0.99
35	Logarithms	\log_{2} p	1.00	1.00	1.00	1.00
36	Polynomial	x^{2}-4x-5	1.00	1.00	1.00	1.00
37	Polynomial	x^{3} + 2x^{2} - 2	0.80	0.70	1.00	0.83
38	Polynomial	x^{4}+2x^{3}-8x^{2}+x-2	1.00	1.00	1.00	1.00
39	Polynomial	x^{4}+6x^{3}+2ax^{2}+bx-3a	1.00	1.00	1.00	1.00
40	Polynomial	a^{4}-7a^{2}b^{2}+kb^{4}	1.00	1.00	1.00	1.00
41	Surds	\sqrt {x}	1.00	0.83	1.00	1.00
42	Surds	\sqrt{1+4x}	1.00	0.90	1.00	0.96
43	Surds	\sqrt{a+b \sqrt{3}}	1.00	1.00	1.00	1.00
44	Surds	\int \sqrt{4x+5} \mathrm{d}x	1.00	1.00	1.00	1.00
45	Surds	\sqrt{x + 1}	1.00	1.00	1.00	1.00
46	Trigonometry	\cos x	1.00	0.83	1.00	1.00
47	Trigonometry	63\sin x+16\cos x	1.00	0.90	1.00	0.96
48	Trigonometry	\sin x \cos x	1.00	1.00	1.00	1.00
49	Trigonometry	\int \sqrt{4x+5} \mathrm{d}x	1.00	1.00	1.00	1.00
50	Trigonometry	\sqrt{x + 1}	1.00	1.00	1.00	1.00
51	Series	\sum x	1.00	1.00	1.00	1.00
52	Series	\sum{r = 1}^{n} r^{2}	1.00	0.89	1.00	1.00

APPENDIX IV BITMAP PRE-PROCESSING

AIV.1 Pre-processing Image URI into Bitmap

```
@Background(serial="ocr")
public void getBitmapFromUri(Uri imageUri) {
    Logger.d("Getting resized bitmaps...");
    BitmapFactory.Options opts = new BitmapFactory.Options();

    try {
        // Get real size
        InputStream input = context.getContentResolver().openInputStream(imageUri);
        opts.inJustDecodeBounds = true;
        opts.inPreferredConfig = Bitmap.Config.ARGB_8888;
        BitmapFactory.decodeStream(input, null, opts);
        input.close();
        int originalWidth = opts.outWidth;
        int originalHeight= opts.outHeight;
        if ((originalWidth == -1) || (originalHeight == -1))
            return;

        // Down sampling
        int originalSize = (originalHeight > originalWidth) ? originalHeight : originalWidth;
        double ratio = (originalSize > MAX_RESCALED_SIZE) ?
                (originalSize/MAX_RESCALED_SIZE) : 1.0;
        opts.inJustDecodeBounds = false;
        opts.inSampleSize = getPowerOfTwoForSampleRatio(ratio);

        // Scaling
        opts.inJustDecodeBounds = false;
        input = context.getContentResolver().openInputStream(imageUri);
        Bitmap bitmap = BitmapFactory.decodeStream(input, null, opts);
        input.close();

        mListener.onBitmapReady(bitmap);

    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
}

private int getPowerOfTwoForSampleRatio(double ratio){
    int k = Integer.highestOneBit((int)Math.floor(ratio));
    if(k==0) return 1;
    else return k;
}
```

AIV.2 Final Bitmap Pre-processing using Leptonica

```
@Background(serial="ocr")
@Override
public void preprocess(Bitmap bitmap) {
    Pix pixs = ReadFile.readBitmap(bitmap);

    pixs = AdaptiveMap.pixContrastNorm(pixs);
    pixs = AdaptiveMap.backgroundNormMorph(pixs);
    pixs = Binarize.sauvolaBinarizeTiled(pixs);
    double angle = Skew.findSkew(pixs);
    pixs = Rotate.rotate(pixs, (float) angle);

    mListener.onBitmapPreprocessed(WriteFile.writeBitmap(pixs));
}
```